VŠB TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Nástroj pro podporu ASPICE hodnocení

ASPICE Assessment Supporting Tool

Bc. Filip Stolár

Diplomová práce

Vedoucí práce: Ing. Svatopluk Štolfa, Ph.D.

Ostrava, 2021

Abstrakt

Předmětem uvedené diplomové práce je seznámení se s normou Automotive SPICE a poté vytvoření nástroje, který umožní uživatelům daného nástroje vytvářet reporty ve formátu PDF. Nástroj umožňuje vkládat libovolná data, avšak hlavním zdrojem dat by měly být výsledky hodnocení procesů z oblasti Automotive. Výsledný nástroj je popsán požadavky, několika případy užití, architekturou a implementací. Na závěr jsou zobrazeny a popsány ukázky naimplementovaného nástroje a jeho užití.

Klíčová slova

Automotive SPICE, Nástroj pro tvorbu reportů, Víceuživatelská aplikace

Abstract

The main subject of the presented master thesis is to become acquainted about Automotive SPICE standard and then a development of support tool which will allow its users to create reports in PDF format. Any data can be imported to the implemented tool but the main data source should be results of Automotive process assessment. Support tool will be described in terms of requirements, use cases, architecture and implementation. At the end of the master thesis are shown samples and pictures of developed support tool.

Keywords

Automotive SPICE, Report creation tool, Multiuser application

Poděkování

Rád bych na tomto místě poděkoval všem, kteří mi s prací pomohli, protože bez nich by tato práce nevznikla. Zejména bych chtěl poděkovat svému vedoucímu panu doktorovi Štolfovi za jeho věcné rady a připomínky. Dále bych chtěl poděkovat své rodině a přítelkyni za podporu.

Obsah

Seznam použitých zkratek a symbolů

SPICE	–	Software Process Improvement and Capability Determination
ASPICE	–	Automotive Software Process Improvement and Capability Determination
IT	–	Informační technologie
IEC	–	International Electrotechnical Commission
PAM	–	Process Assessment Model
PRM	–	Process Reference Model
RFP	–	Request for proposal
ACQ	–	Acquisition process group
SPL	–	Supply process group
SYS	–	System engineering process group
SWE	–	Software engineering process group
SUP	–	Supporting process group
MAN	–	Management process group
PIM	–	Process improvement process group
REU	–	Reuse process group
PDF	–	Portable Document Format
ISO	–	International Organization for Standardization
IEC	–	International Electrotechnical Commission
tzv.	–	takzvaný
XLSX	–	Přípona souborů specifikace Office Open XML vytvořených v aplikaci Microsoft Excel
BP	–	Base practices
WP	–	Work products
GP	–	General practice
GR	–	General resource
MVC	–	Model View Controller
URL	–	Uniform Resource Locator

SOAP	–	Simple Object Access Protocol
XML	–	Extensible Markup Language
REST	–	Representational State Transfer
JSON	–	Javascript Object Notation
SVG	–	Scalable Vector Graphics
API	–	Application Programming Interface
SQL	–	Structured Query Language
NoSQL	–	None - Structured Query Language
ACID	–	Atomicity, Consistency, Isolation, Durability
ID	–	Identifier
HTTP	–	Hypertext Transfer Protocol
HTML	–	Hypertext Markup Language

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

Hodnocení procesů, dle standardu Automotive SPICE, slouží výrobcům automobilů k volbě vhodného dodavatele softwarových komponent, systémů pro správu dat a služeb pro integraci softwaru a hardwaru.

Práce bude zahrnovat nastudování problematiky hodnocení ASPICE a tvorbu nástroje, který by umožňoval hodnotitelům procesů vložit do tohoto nástroje data s výsledky hodnocení a z těchto dat vytvářet uživatelem definované reporty. Nástroj bude poskytovat víceuživatelský přístup a tedy nezávislost na operačním systému, ve kterém uživatel bude využívat daný nástroj. Toho bude dosaženo díky naimplementované webové aplikaci.

V kapitole 2 bude nejprve popsána norma SPICE, ze které vychází norma Automotive SPICE. V popisu normy Automotive SPICE budou zmíněny principy a modely, dle kterých probíhá hodnocení procesů v rámci zvolené organizace probíhá. V poslední části druhé kapitoly budou popsány nástroje, které slouží k hodnocení procesů. Na trhu dosud nebyl nalezen žádný volně dostupný nástroj, který by k tomuto účelu sloužil a proto budou uvedeny alespoň některé komerční nástroje. Výstup z nástroje Capability Adviser ve formátu souboru xlsx bude poskytnut vedoucím této diplomové práce. Tento výstup slouží jako testovací data pro vznikající nástroj pro reportování.

Kapitola 3 bude popisovat požadavky na vznikající reportovací nástroj. V první části požadavků budou nejprve rozepsány jednotlivé případy užití, které budou poté následovány návrhem uživatelského rozhraní webové aplikace.

Kapitola 4 bude popisovat architekturu vzniklého nástroje pro reportování a bude mít za cíl popsat celý systém a jmenovat použité technologie a vzory. Popsána bude databáze, klient i server. Nebudou chybět ani náhledy na systém pomocí diagramů.

Kapitola 5 se bude věnovat samotné implementaci nástroje pro reportování. Zde bude detailně popsána vzniklá serverová část i webová aplikace.

V poslední kapitole 6 bude zobrazena výsledná naimplementovaná aplikace pro tvorbu reportů, kde budou prezentovány jednotlivé části webové aplikace.

Kapitola 2

Automotive SPICE a hodnocení procesů

V této kapitole je nejprve popsán standard ISO/IEC 15504 a poté varianta tohoto standardu, která je specifikována pro automobilový průmysl.[1] Jsou zde také popsány principy a metodiky referenčního a hodnotícího modelu.

2.1 SPICE

Mezinárodní standard ISO/IEC 15504 (SPICE)[2] je referenční model pro hodnocení a posuzování business procesů, které se zpravidla zaměřují na vývoj softwaru. Jádro této normy je zaměřeno na zlepšování procesů ve vlastní organizaci a stanovení způsobilosti procesu (úroveň způsobilosti). Hodnocení procesu se provádí za pomocí dvourozměrného referenčního a hodnotícího modelu. Dimenze procesu slouží k identifikaci a výběru procesů, které budou při hodnocení zkoumány. Druhou dimenzí je stupeň odbornosti neboli způsobilosti, která se využívá k vyhodnocení příslušné schopnosti provádět proces.[3]

2.1.1 Dimenze procesu

Procesní dimenze definuje procesy, které jsou rozděleny do pěti podkategorií[4]:

- zákazník-dodavatel
- inženýrství
- podpora
- management
- organizace

S každou nově vydanou částí normy se kategorie procesů rozšiřují o procesy zaměřené na IT služby a podnikové procesy.

2.1.2 Úrovně způsobilosti a procesní atributy

Pro každý proces definuje standard ISO/IEC 15504 úroveň způsobilosti na základě následujícího měřítka[4]:

- 5 - Optimizing process (Optimalizace procesu) - Proces je předvídatelný, proto je možné jej neustále přizpůsobovat změnám v organizaci.
- 4 - Predictable process (Předvídatelný proces) - Proces splňuje požadavky nižší úrovně způsobilosti a má předem vymezené hranice, ve kterých tento proces operuje. V případě, že jsou nalezeny nějaké odchylky od těchto vymezených hranic, pak může dojít k nápravným akcím, které mají za úkol dané odchylky zaznamenat.
- 3 - Established process (Zavedený proces) - Proces je implementován a splňuje všechny nároky nižšího stupně způsobilosti, navíc však dosáhne všech požadovaných výsledků.
- 2 - Managed process (Řízený proces) - Proces, který byl implementován a splňuje úroveň provedeného procesu, je monitorován, řízen a může být přizpůsoben změnám.
- 1 - Performed process (Provedený proces) - Proces, který je implementován splňuje svůj účel.
- 0 - Incomplete process (Neúplný proces) - Proces není implementován vůbec nebo nesplňuje svůj účel.

Způsobilost procesu je měřena pomocí procesních atributů. Mezinárodní standard definuje devět procesních atributů[4]:

- 1.1 Process performance (Výkonnost procesu)
- 2.1 Performance management (Řízení výkonu)
- 2.2 Work product management (Řízení pracovního produktu)
- 3.1 Process definition (Definice procesu)
- 3.2 Process deployment (Nasazení procesu)
- 4.1 Process measurement (Měření procesu)
- 4.2 Process control (Řízení procesu)
- 5.1 Process innovation (Inovace procesu)
- 5.2 Process optimization (Optimalizace procesu)

Každý procesní atribut se skládá z několika postupů, které jsou převedeny do indikátorů, jenž napomáhají se samotným hodnocením.

2.1.3 Měřítko pro hodnocení procesních atributů

Každý procesní atribut je ohodnocen následujícím, čtyř bodovým, měřítkem[4]:

- Not achieved (0-15%) - Nedosaženo - Neexistuje žádný důkaz nebo je důkaz nedostačující pro dosažení daného procesního atributu hodnoceného procesu.
- Partially achieved (>15-50%) - Částečně dosaženo - Existuje důkaz o přístupu a dosažení daného procesního atributu v hodnoceném procesu. Některé aspekty dosažení procesního atributu mohou však být nepředvídatelné.
- Largely achieved (>50–85%) - Dosaženo z velké části - Existují důkazy o systematickém přístupu k procesnímu atributu a jeho výrazném úspěchu. Mohou však existovat slabiny, které souvisí právě s daným procesním atributem.
- Fully achieved (>85–100%) - Plně dosaženo - Existuje jasný důkaz o úplném splnění a systematickém přístupu k danému procesnímu atributu. Pokud se vyskytnou v rámci hodnocení nějaké slabiny, pak jsou zanedbatelné.

Hodnocení je založeno na důkazech, které byly shromážděny pomocí postupů popsaných ve způsobilostech procesu.[5] Tyto důkazy pak vedou k odhalení skutečné hodnoty zkoumaných procesních atributů.

2.1.4 Proces hodnocení

Detailní hodnocení procesů je popsáno v normě ISO/IEC 15504, a to v části druhé a třetí.[4] Jedním z požadavků je použití konformní metodiky k hodnocení procesů. Tato metodika není přímo specifikována standardem, ale definována požadavky, které má metodika hodnocení splňovat. Stanovuje také požadavky pro vývojáře této metodiky a samotné hodnotitele procesů.

Standard poskytuje obecnou příručku hodnotitelům, která musí být podpořena formálním školením a vedením zaměřeným na detaily během prvních hodnocení.[6] Proces hodnocení může být zobecněn do následujících kroků: Zahájení hodnocení, výběr hodnotitele a hodnotícího týmu, plán hodnocení spolu s určením procesů a hodnocených organizačních jednotek, instruktáž před zahájením hodnocení, sběr dat, validace dat, ohodnocení procesu a na závěr reportování výsledků hodnocení.

2.1.5 Kvalifikace a kompetence hodnotitelů

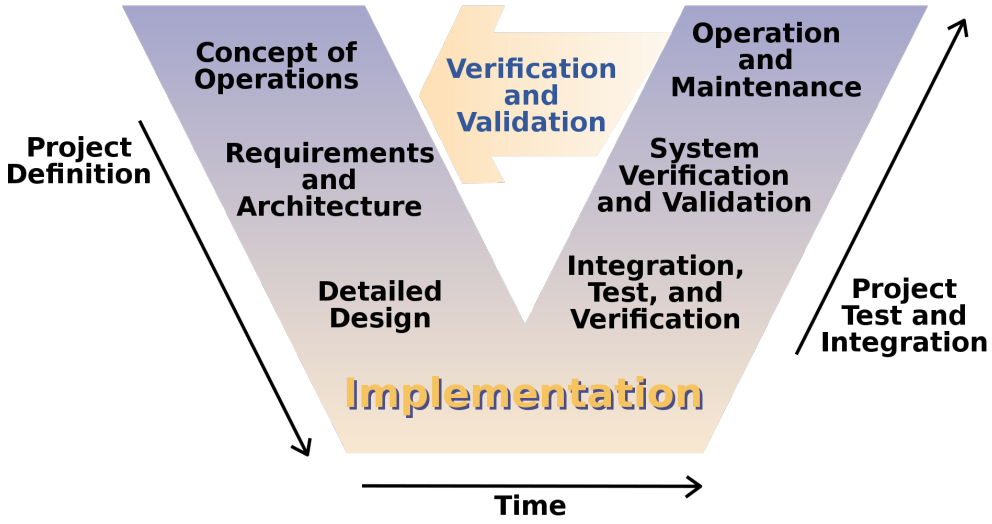
Hodnocení procesů je činnost, při které organizace dostává hodnocení, na jehož základě může získávat další zakázky na výrobu softwaru. Pro úspěšné a přesné hodnocení je tedy vyžadována určitá úroveň kvalifikace a zkušeností hodnotitelů.[1] Tyto zkušenosti zahrnují komunikační dovednosti, příslušné vzdělání a zaškolení.

2.2 ASPICE

2.2.1 Úvod

Automotive Software Process Improvement and Capability dEtermination (ASPICE) neboli specifikace sloužící ke zlepšování procesů a způsobilosti procesů pro automobilový průmysl.[7] Vychází ze standardu ISO/IEC 15504, který byl popsán výše. Hlavní úlohou tohoto standardu je poskytnout pokyny ke zlepšení procesu vývoje softwaru a hodnocení dodavatelů.

Standard definuje dva modely - PRM (referenční model procesu) a PAM (model hodnocení procesu). Tento standard využívá tzv. V-Model (obrázek 2.1), ve kterém ke každému procesu od požadavků ke zdrojovému kódu existuje korespondující test.



Obrázek 2.1: V-Model procesů systémového inženýrství[8]

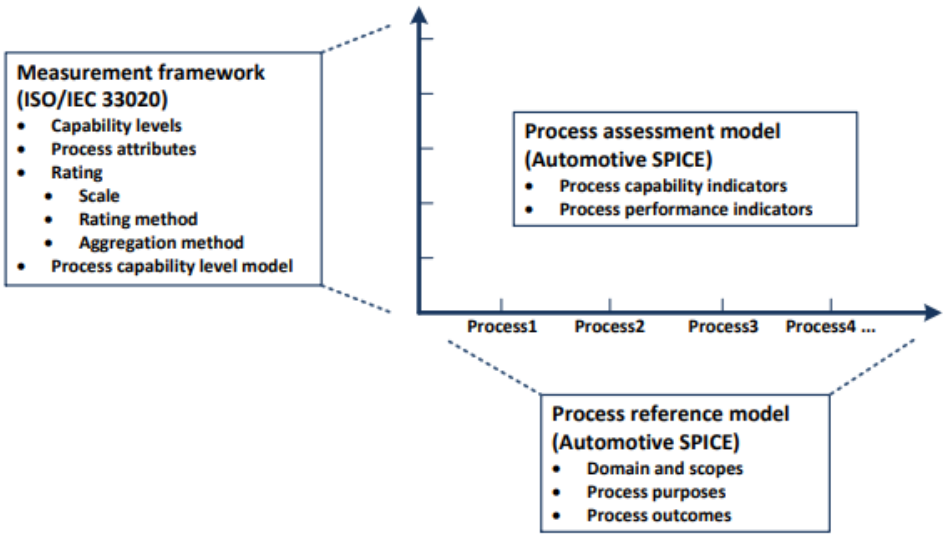
Hlavní myšlenkou tohoto modelu je sled následujících procesů[7]:

- Sběr požadavků od zákazníka. Tyto požadavky mohou být nesrozumitelné, a proto je cílem tyto požadavky upřesnit.
- Projekt by měl převádět požadavky od zákazníka na systémové požadavky.
- Systémový architekt rozčlení požadavky do logických částí. Součástí je také návrh hardwaru, softwaru a komunikace mezi těmito komponenty.
- Pro každou softwarovou službu se odvodí softwarové požadavky ze systémových požadavků.
- Softwarový architekt rozčlení softwarové požadavky na softwarové jednotky.
- Každá takto vzniklá softwarová jednotka je navržena a poté implementována.

- Implementace unit testů, kontrola zda implementace koresponduje s návrhem a kontrola nefunkčních požadavků.
- Integrační testy nad softwarovou architekturou.
- Kvalifikační testy nad softwarovými požadavky.
- Integrační testy nad systémovou architekturou.
- Kvalifikační testy nad systémovou architekturou.
- Akceptační testy, které jsou provedeny zákazníkem.

2.2.2 Stanovení způsobilosti procesu

Koncept stanovení způsobilosti procesu použitím PAM (Process Assessment Model) je založen na dvoudimenzionálním frameworku (obrázek 2.2). Podobný koncept je obsažen v normě SPICE. První dimenze tohoto frameworku se skládá z procesů definovaných v PRM (Process Reference Model). Tato dimenze se označuje také jako procesní dimenze. Druhá dimenze (dimenze způsobilosti) je složena z úrovní způsobilosti, které jsou dále rozděleny na procesní atributy, jež poskytují měřitelné charakteristiky způsobilosti procesu[9].



Obrázek 2.2: Vztah v modelu hodnocení procesů[9]

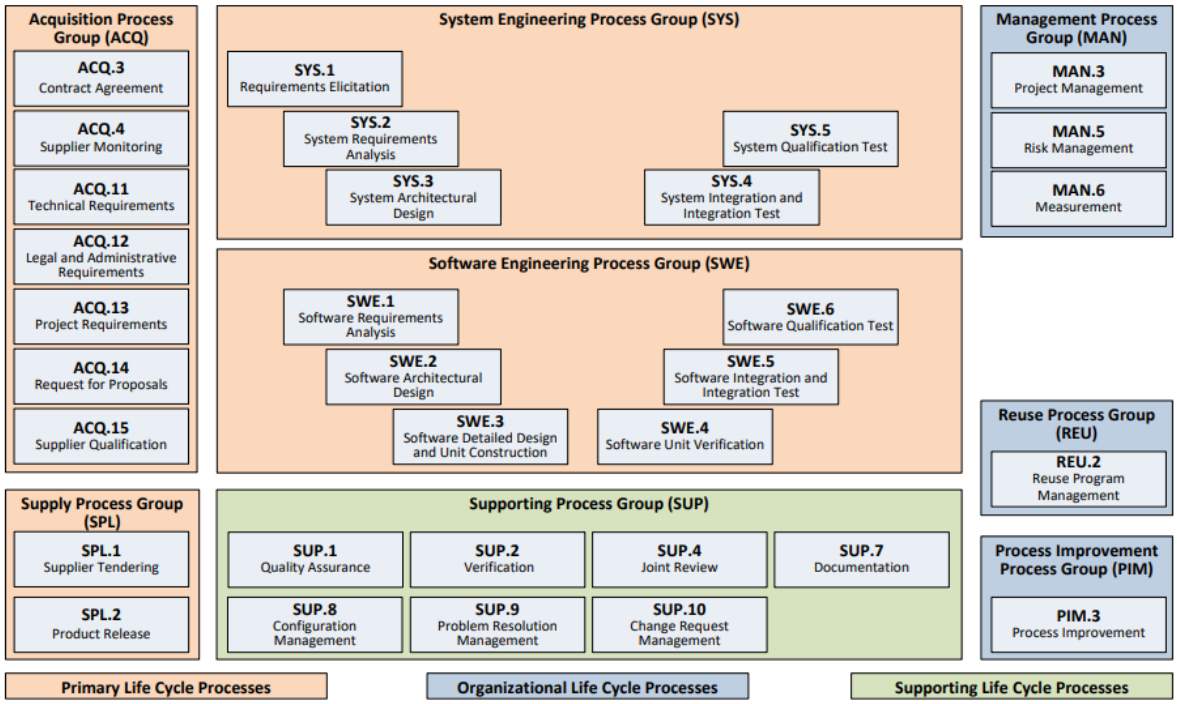
2.2.3 Referenční model procesu - PRM

Procesy jsou seskupeny do 3 hlavních kategorií a ty jsou dále rozděleny do skupin dle typů procesů (obrázek 2.3). Existují tři kategorie procesů[9]:

Hlavní procesy životního cyklu (Primary Life Cycle Processes)

Organizační procesy životního cyklu (Organizational Life Cycle Processes)

Podpůrné procesy životního cyklu (Supporting Life Cycle Processes)



Obrázek 2.3: Referenční model procesu Automotive SPICE[9]

2.2.4 Hlavní procesy životního cyklu

Hlavní procesy životního cyklu se skládají z procesů, které mohou být použity zákazníkem při přebírání produktů od dodavatele nebo samotným dodavatelem při dodávce produktů zákazníkovi. Tyto procesy zahrnují dokumenty a postupy, které jsou potřebné ke specifikaci, návrhu, vývoji, integraci a testování[9]. Kategorie procesů hlavního životního cyklu obsahuje tyto podskupiny[9]:

- Skupina akvizičních procesů
- Skupina procesů pro dodávání

- Skupina procesů systémového inženýrství
- Skupina procesů softwarového inženýrství

Skupina akvizičních procesů (ACQ) je složena z procesů, které jsou vykonávány zákazníkem nebo dodavatelem, pokud je sám v roli zákazníka, pro své vlastní dodavatele, za účelem získání produktu nebo služby[9].

ACQ.3 - Podepsání smlouvy

ACQ.4 - Monitorování dodavatele

ACQ.11 - Technické požadavky

ACQ.12 - Právní a administrativní požadavky

ACQ.13 - Projektové požadavky

ACQ.14 - RFP (Requests for proposals) nabídky

ACQ.15 - Kvalifikace dodavatele

Skupina procesů pro dodávání (SPL) obsahuje procesy vykonávané dodavatelem za účelem dodání produktu nebo služby[9].

SPL.1 - Výběrové řízení na dodavatele

SPL.2 - Vydání produktu

Skupina procesů systémového inženýrství (SYS) se skládá z procesů zaměřených na sběr a správu požadavků od zákazníka nebo od interních týmu v organizaci. Dále je složena z procesů, které definují systémovou architekturu, integraci a testování na úrovni systému[9].

SYS.1 - Sběr požadavků

SYS.2 - Analýza systémových požadavků

SYS.3 - Návrh systémové architektury

SYS.4 - Systémová integrace a integrační testování

SYS.5 - Systémové kvalifikační testy

Skupina procesů softwarového inženýrství (SWE) je složena z procesů zaměřujících se na správu softwarových požadavků získaných ze systémových požadavků a procesů, které slouží k vývoji související softwarové architektury a návrhu, a také k integraci a testování softwaru[9].

SWE.1 - Analýza softwarových požadavků

SWE.2 - Návrh architektury systému

SWE.3 - Detailní návrh softwaru a sestrojení softwarových jednotek

SWE.4 - Verifikace softwarové jednotky

SWE.5 - Softwarová integrace a integrační testy

SWE.5 - Softwarové kvalifikační testy

2.2.5 Podpůrné procesy životního cyklu

Procesy životního cyklu podpory (SUP) se skládají z procesů, které mohou být využity jinými procesy v rámci životního cyklu[9].

SUP.1 - Ověřování kvality

SUP.2 - Verifikace

SUP.4 - Společné přezkoumání

SUP.7 - Dokumentace

SUP.8 - Správa konfigurací

SUP.9 - Správa řešení problémů

SUP.10 - Správa požadavků na změnu

2.2.6 Organizační procesy životního cyklu

Organizační procesy životního cyklu zahrnují procesy, které vyvíjí procesní, produktová a zdrojová aktiva, která mohou napomoci k dosažení obchodních cílů organizace. Organizační procesy životního cyklu obsahují následující podskupiny[9]:

- Skupina procesů pro správu
- Skupina procesů ke zlepšování procesů
- Skupina procesů pro znovupoužitelnost

Skupina procesů pro správu (MAN) řadí procesy, které mohou být použity kýmkoliv, kdo spravuje projekt nebo proces, v rámci životního cyklu[9].

MAN.3 - Správa projektů

MAN.5 - Správa rizik

MAN.6 - Měření

Skupinu procesů ke zlepšování procesů (PIM) pokrývá pouze jeden proces, který obsahuje praktiky ke zlepšení procesů vykonávaných nějakou organizační jednotkou[9].

PIM.3 - Zlepšení procesu

Skupinu procesů pro znovupoužitelnost (REU) pokrývá rovněž jeden proces, tak jak je tomu u předchozí skupiny. Daný proces slouží k vyhledávání možností, jež by se daly využít v rámci organizace. Tento postup šetří čas, kdy by se musely znovu vyvíjet aktiva, která již byla vyvinuta[9].

REU.2 - Správa programu pro znovupoužitelnost

2.2.7 Framework pro měření způsobilosti

Framework pro měření způsobilosti poskytuje nezbytně nutné požadavky a pravidla pro dimenzi způsobilosti. Dále popisuje schéma, které umožňuje hodnotitelům stanovit úroveň způsobilosti konkrétního procesu. Úrovně způsobilosti, jejichž popis je také součástí tohoto frameworku, budou popsány později[9].

Za účelem možného hodnocení procesů, framework specifikuje takzvané procesní atributy, které popisují měřitelné vlastnosti úrovně způsobilosti. Každému procesnímu atributu je přiřazena charakteristická úroveň způsobilosti. Pravidla, podle kterých hodnotitel určí výslednou úroveň způsobilosti procesu, jsou zastoupena v modelu úrovní způsobilosti[9].

2.2.8 Úrovně způsobilosti procesů a procesních atributů

Úrovně způsobilosti a procesních atributů jsou totožné s těmi, které jsou definovány v normě ISO/IEC 33020 (procesní hodnocení). Procesní atributy jsou tedy vlastnosti procesu, které jsou vyhodnoceny na stupnici úspěšnosti, která poskytuje míru způsobilosti[9]. Tyto vlastnosti jsou použitelné na všechny typy procesů. Úroveň způsobilosti je množinou procesních atributů, jež společně zlepšují schopnost provádět proces. Také představují rozumnou cestu k vylepšování způsobilosti kteréhokoli procesu[9]. Úrovně způsobilosti se téměř shodují s těmi, které byly definovány v úvodu SPICE, proto zde nebudou znovu samostatně uváděny. Jediná drobná odchylka spočívá v názvu poslední páté úrovně, kdy je optimalizující proces nahrazen procesem inovovaným. Významově se však jedná o stejnou úroveň způsobilosti. V níže uvedené tabulce (tabulka 2.1) jsou popsány procesní atributy, které výslovně určují, do které úrovně způsobilosti daný proces spadá[9].

Úroveň 0 - Neúplný proces	Neexistují žádné procesní atributy.
Úroveň 1 - Provedený proces	PA 1.1 - Procesní atribut výkonnosti procesu.
Úroveň 2 - Řízený proces	PA 2.1 - Procesní atribut řízení výkonnosti procesu. PA 2.2 - Procesní atribut výkonnosti pracovního produktu.
Úroveň 3 - Zavedený proces	PA 3.1 - Procesní atribut definice procesu. PA 3.2 - Procesní atribut nasazení procesu.
Úroveň 4 - Předvídatelný proces	PA 4.1 - Procesní atribut kvantitativní analýzy. PA 4.2 - Procesní atribut kvantitativního řízení.
Úroveň 5 - Inovovaný proces	PA 5.1 - Procesní atribut inovace procesu. PA 5.2 - Procesní atribut implementace inovací procesu.

Tabulka 2.1: Tabulka procesních atributů přiřazených k jednotlivým úrovním způsobilosti.[9]

2.2.9 Hodnocení procesních atributů

Již dříve byl zmíňený framework měření způsobilosti, který má poskytovat sadu požadavků a pravidel z hlediska dimenze způsobilosti. Měřítko hodnocení stanovené frameworkem je možné detailněji popsat. Umožněna je také změna metodiky hodnocení v závislosti na kategorii hodnocení[9]. Definice měřítka hodnocení procesních atributů je chápána i jako dosažená míra způsobilosti procesního atributu hodnoceného procesu. Popis tohoto měřítka již byl uveden v podkapitole 2.1.3, avšak níže je uvedeno rozšíření o detailnější specifikaci[9]:

- **P - Částečně dosaženo (Partially achieved)** - Procento dosažení tohoto hodnocení je v rozmezí od více než 15% do 32.5% včetně. Existuje důkaz o přístupu a dosažení daného procesního atributu v hodnoceném procesu. Mnohé aspekty dosažení procesního atributu jsou však nepředvídatelné.

- **P + Částečně dosaženo (Partially achieved)** - Procento dosažení tohoto hodnocení je v rozmezí od více než 32.5% do 50% včetně. Existuje důkaz o přístupu a dosažení daného procesního atributu v hodnoceném procesu. Některé aspekty dosažení procesního atributu mohou však být nepředvídatelné.

- **L - Dosaženo z velké části (Largely achieved)** - Procento dosažení tohoto hodnocení je v rozmezí od více než 32.5% do 50% včetně. Existují důkazy o systematickém přístupu k

procesnímu atributu a jeho výraznému dosažení. Skrývá v sobě mnoho slabin, které souvisí s daným procesním atributem.

- **L + Dosaženo z velké části (Largely achieved)** - Procento dosažení tohoto hodnocení je v rozmezí od více než 67.5% do 85% včetně. Existují důkazy o systematickém přístupu k procesnímu atributu a jeho výrazném dosažení. Avšak mohou existovat slabiny, které souvisí právě s daným procesním atributem.

2.2.10 Model úrovní způsobilosti procesu

Úroveň způsobilosti procesu dosažená určitým procesem by měla být odvozena na základě hodnocení procesních atributů, kde toto odvození je znázorněno na obrázku 2.4. Model úrovní způsobilosti procesu udává kritéria, která určují dosažení konkrétní úrovně způsobilosti v závislosti na hodnocení procesních atributů.[9] Hlavním kritériem dosažení specifické úrovně je splnění všech procesních atributů dané úrovně minimálně na hodnocení L (Dosaženo z velké části) a hodnocení F (Plně dosaženo) u všech atributů, které jsou obsaženy ve všech nižších úrovních.[9]

Scale	Process attribute	Rating
Level 1	**PA 1.1: Process Performance**	**Largely**
Level 2	PA 1.1: Process Performance	Fully
	PA 2.1: Performance Management	**Largely**
	PA 2.2: Work Product Management	**Largely**
Level 3	PA 1.1: Process Performance	Fully
	PA 2.1: Performance Management	Fully
	PA 2.2: Work Product Management	Fully
	PA 3.1: Process Definition	**Largely**
	PA 3.2: Process Deployment	**Largely**
Level 4	PA 1.1: Process Performance	Fully
	PA 2.1: Performance Management	Fully
	PA 2.2: Work Product Management	Fully
	PA 3.1: Process Definition	Fully
	PA 3.2: Process Deployment	Fully
	PA 4.1: Quantitative Analysis	**Largely**
	PA 4.2: Quantitative Control	**Largely**
Level 5	PA 1.1: Process Performance	Fully
	PA 2.1: Performance Management	Fully
	PA 2.2: Work Product Management	Fully
	PA 3.1: Process Definition	Fully
	PA 3.2: Process Deployment	Fully
	PA 4.1: Quantitative Analysis	Fully
	PA 4.2: Quantitative Control	Fully
	PA 5.1: Process Innovation	**Largely**
	PA 5.2: Process Innovation Implementation	**Largely**

Obrázek 2.4: Model úrovní způsobilosti[9]

2.2.11 Model hodnocení - PAM

Tento model poskytuje indikátory k identifikaci, zda jsou výstupy procesu a procesních atributů zahrnuty v již zaběhnutém procesu organizační jednotky. Tyto indikátory poskytují oporu pro hodnotitele při sběru nezbytných důkazů, které by podpořily výsledné hodnocení[9]. Nejsou však povinnou součástí.

Za účelem posouzení, zda jsou výstupy procesů zahrnuty, jsou potřebné objektivní důkazy. Veškeré tyto důkazy pochází ze zkoumání pracovních produktů, obsahu dokumentace hodnoceného procesu, a také z posudků od samotných manažerů a vykonavatelů daného procesu. Veškeré tyto objektivní důkazy jsou namapovány na indikátory modelu hodnocení, aby umožnily vytvoření popisu k důležitým procesním výstupům a cílům procesních atributů[9].

Tyto indikátory jsou dvojího typu. První z nich jsou tzv. *indikátory* výkonnosti procesu, které se aplikují pouze na úrovni způsobilosti 1. Mají za cíl odhalit rozsah splněných procesních výstupů. Jinými slovy se jedná o zjištění, které výstupy procesu jsou splněny. Druhým typem indikátorů jsou tzv. indikátory způsobilosti procesu, které se aplikují na úrovně způsobilosti od 2 do 5. Poskytují informace o splnění nebo nesplnění procesních atributů[9].

Indikátory hodnocení jsou mechanismem, který odhaluje, zda byly použity některé vybrané praktiky, jež popisují důkazy získané během hodnocení. Důkazy zaznamenané během hodnocení by měly být vytvářeny formou, která navazuje na procesní indikátory. Tato praktika má za cíl podpořit posudek hodnotitele[9].

2.2.12 Indikátory výkonnosti

Existují dva typy indikátorů výkonnosti[9]:

- Základní praktiky (BP - Base practices)
- Pracovní produkty (WP - Work products)

Oba typy indikátorů jsou propojeny s jedním nebo více výstupy procesu. Každý z indikátorů, spadající do jednoho z těchto dvou typů, je specifický pro konkrétní proces a není tedy obecně definován. Indikátory BP jsou zaměřeny na specifické aktivity, kdežto indikátory WP jsou zaměřeny na výsledky.

Obecně model hodnocení poskytuje popis pro každý pracovní produkt. Tento popis slouží jako pomůcka pro hodnotitele. Je rychlým a spolehlivým informačním zdrojem během hodnocení. Ukazuje, jak má obecně vypadat struktura hodnocení, avšak dává hodnotitelům možnost mít tuto strukturu mírně odlišnou. Ve skutečnosti by měl být popis pracovních produktů a dokumentů v organizaci poskytnutý samotnou organizací. Organizace by se měla ujistit, že pracovní produkty splňují svůj účel i potřebu a podporují cyklus vývoje softwaru.[9]

2.2.13 Indikátory způsobilosti

Indikátory způsobilosti jsou tyto[9]:

- Obecná praktika (GP - general practice)
- Obecný zdroj (GR - general resource)

Indikátory způsobilosti jsou propojeny s výsledky hodnocení a oproti indikátoru výkonnosti jsou popsány obecně. První z typů, tedy GP, obsahuje indikátory, které se zaměřují zpravidla na specifické činnosti. Druhý z typů, tedy GR, obsahuje indikátory orientované na infrastrukturu. Hodnotitel musí shromáždit data a sepsat důkazy, které podporují indikátory způsobilosti během hodnocení. Oba dva typy, GP i GR, jsou alternativní ukazatele, jež může hodnotitel využít[9].

Pojem proces může být vnímán na různých úrovních abstrakce v rámci modelu hodnocení. První z abstrakcí je samotný model způsobilosti, který popisuje cíle daného procesu. Tato úroveň abstrakce poskytuje odpovědi na následující otázky[9]:

- Co má být provedeno?
- Proč to má být provedeno?
- Jaké jsou technické závislosti?

Druhou abstrakcí jsou metody, které popisují jakým způsobem daného cíle dosáhnout.[9]

- Metody, nástroje, šablony, metriky
- Definice, postupy
- Určení autorit a zodpovědností

Poslední abstrakcí je samotné provedení, které popisuje konkrétní činnosti během cesty k cíli.[9]

- Vytvoření postupu
- Nastavení
- Výkonnost vytvořeného postupu

2.3 Nástroje pro hodnocení procesů

Na trhu nebyl dosud nalezen volně dostupný, relevantní software, který by umožňoval vkládat data z hodnocení procesů a z nich generovat uživatelem sestavené reporty na základě definovaných vstupů a objektů od uživatele a navíc byl určen přímo pro hodnotitele procesů. Na trhu však existuje několik komerčních nástrojů, které slouží k samotnému hodnocení procesů a jsou naimplementovány tak, aby obsahovaly vše potřebné co je definováno normou ASPICE. Níže budou v krátkosti představeny tři nástroje, které k tomuto účelu slouží.

Nástroj: Capability Adviser

Nástroj pro hodnocení procesů Capability Adviser[10] je vyvíjený jako víceuživatelský portál sloužící samotnému týmu hodnotitelů. Hodnotitelé si mohou zobrazit komentáře a hodnocení od jiných hodnotitelů v reálnem čase a mohou také přidávat související důkazy, které mohou být prolinkovány s konkrétním hodnoceným procesem. Tento nástroj umožňuje generování reportů z výsledků hodnocení do formátu xlsx nebo word, ale umožňuje také export reportu do pdf. Capability Adviser nabízí tři řešení, jakým způsobem mohou hodnotitelé mezi sebou sdílet tyto informace.[11] Prvním z nich je serverové řešení, které funguje na principu centrálního prvku (serveru) ve firemním intranetu nebo internetu, na který se připojují jednotliví klienti. Druhým řešením je offline verze, která je dostupná například pro laptopy, kde odpadá možnost vidět hodnocení ostatních hodnotitelů v reálném čase. Poslední řešení je založeno na mikropočítačové platformě Raspberry Pi, která slouží pro externí spolupráci během hodnocení. Nástroj také definuje několik uživatelských rolí, kde každá role má přístup k různým funkcionalitám. Výstupní report z tohoto nástroje ve formátu xlsx byl poskytnut jako testovací data do vznikajícího systému, který bude později popsán v této práci.

Nástroj: SPiCE 1-2-1

SPiCE 1-2-1 je dalším nástrojem[12] pro hodnocení procesů, který se stejně jako předchozí zmíněný nástroj řídí podle ASPICE. Nástroj umožňuje zadávání hodnocení od samotného hodnotitele a poté z finálních výsledků generuje reporty ve formě grafů. Nástroj nabízí tři řešení, které se od sebe liší funkčností a stylem zadávání hodnocení. Prvním z nich je osobní řešení, kde může hodnotitel vytvářet nová hodnocení nebo si otevřít již existující hodnocení a zadávat hodnocení ke konkrétním procesům. Hodnotitel si může také přečíst různé definice a vysvětlivky, které mu napomohou během procesu hodnocení. Druhé řešení je podnikové, které oproti osobnímu řešení disponuje funkcí pro spojení dvou souborů s hodnocením. Posledním řešením je korporátní řešení, jenž může být na míru přizpůsobeno dle požadavků hodnotitelů, kteří si kupují licenci toho nástroje. Oproti předešlému nástroji je tento nástroj zastaralejší a nenabízí možnost klient/server řešení, což nutí hodnotitele mezi sebou sdílet soubory s hodnocením a poté je slučovat, aby se dopracovali k výslednému hodnocení případně grafům.

Nástroj: Callis Trace

Callis Trace[13], je stejně jako popsané předchozí dva nástroje, software sloužící k hodnocení procesů. Podle popisu je software vhodný pro hodnocení, které vykonává více hodnotitelů současně. Software umožňuje vkládat a exportovat vlastní příručky pro hodnotitele či vybrat rozsah hodnocení v rámci plánovaní nějakého hodnocení, což znamená například určení, který hodnotitel bude zastávat jakou roli v průběhu hodnocení. Pomocí tzv. Workspaces umožňuje kooperaci a konsolidaci mezi několika hodnotiteli současně.

Podobně jako nástroj Capability Adviser nabízí několik možností nasazení: centrální server v intranetu nebo jako cloudové řešení a offline aplikaci pro laptopy, kde jednotlivá hodnocení mohou být z offline nástroje nahrány na centrální server. Nástroj poskytuje náhled na existující důkazy a hodnocení buď pohledem seznamu nebo pohledem tabulky. Výstup z finálního hodnocení je možné stáhnout jako report ve formátu souboru Microsoft Word nebo Microsoft Excel. Nástroj také dává možnost zvolit si hodnotící měřítko podle normy SPICE nebo ASPICE. Základní a rozšířené měřítko bylo popsáno v kapitole 2.4.2 pro ASPICE a kapitole 2.13 pro SPICE.

Kapitola 3

Analýza požadavků

V této části práce jsou nejprve rozepsány požadavky na vyvíjený nástroj a poté následuje popis funkčností pomocí diagramu případu užití včetně detailního popisu a návrh grafického rozložení webové aplikace.u

3.1 Požadavky

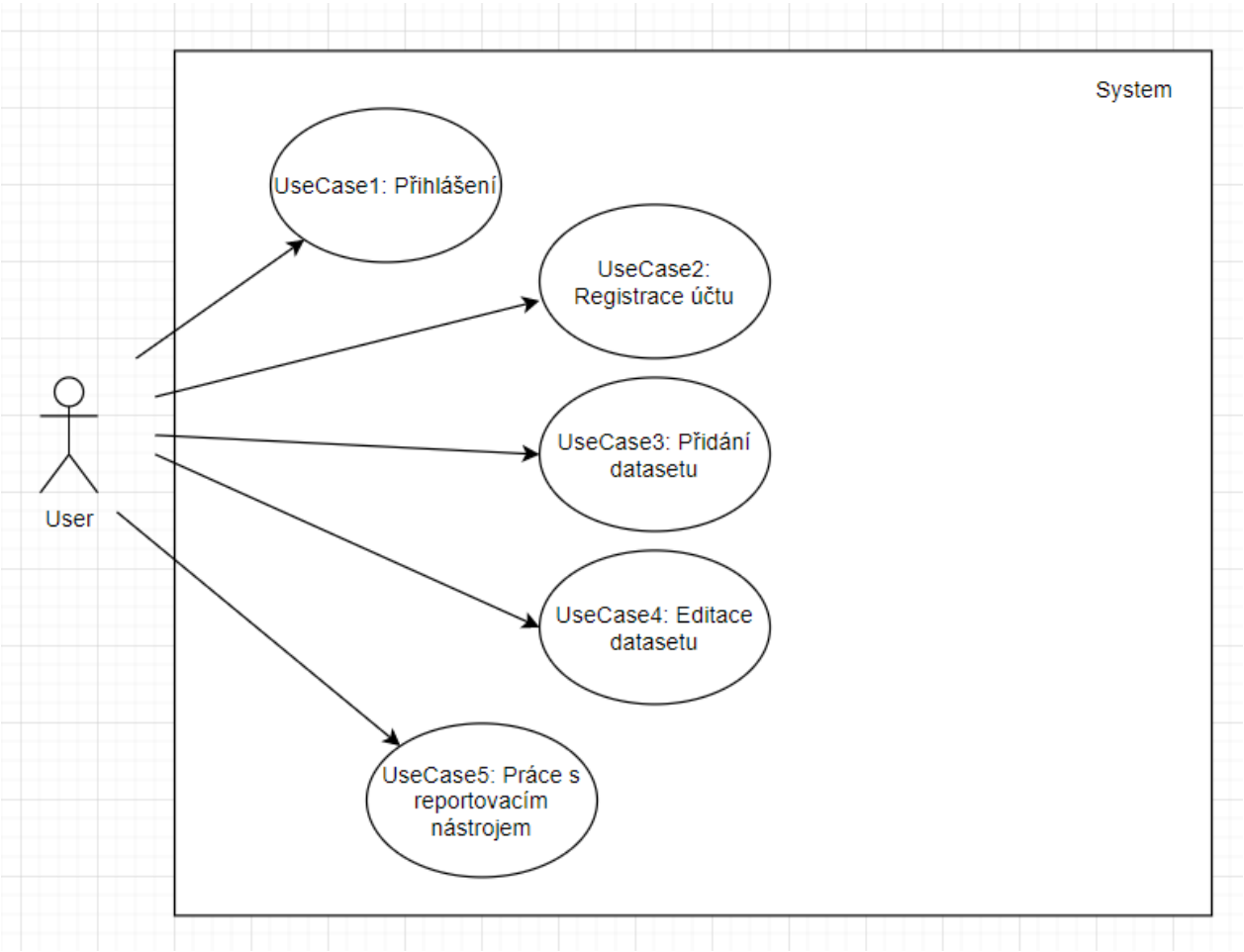
Výsledkem této práce bude systém, který umožní hodnotitelům procesů v dané organizaci pracovat s daty, jež jsou během hodnotícího procesu získána. Tato data jsou dostupná v souboru formátu xlsx. Vstupem do vyvíjeného systému jsou právě tato data v daném formátu. Systém bude umožňovat uživateli vybrat data, nad kterými budou vytvářeny uživatelem definované reporty. Takto vzniklé reporty bude možné stáhnout z implementovaného sytému v souboru ve formátu PDF. Uživatelem definované reporty budou systémem zaznamenávány a mohou být zpětně načteny a opětovně staženy ve formátu PDF. Definované reporty obsahují uživatelem přidané tabulky, grafy a texty. Systém je určen hlavně pro samotné hodnotitele procesů.

3.2 Případy užití

Na obrázku 3.1 je zobrazen diagram případů užití, který definuje všechny scénáře, které systém obsahuje. Systém se skládá z následujících scénářů, které budou popsány později:

1. UseCase1: Příhlášení. Příhlášení je tvořeno formulářem, kde uživatel zadá svůj zaregistrovaný email a heslo. V případě úspěchu je uživatel úspěšně přihlášen do systému.

2. UseCase2: Registrace. Registrace je součástí formuláře pro přihlášení. Pro zaregistrování nového uživatele musí být ve formuláři vyplněno jméno, email a heslo. Uživatel je schopen přepínat se mezi přihlášením a registrací nového účtu. V případě úspěšné registrace je uživateli vytvořen příslušný účet.



Obrázek 3.1: Diagram případů užití vytvářeného systému.

3. UseCase3: Přidání datasetu. Funkce přidání nového datasetu je tvořena formulářem, kde přihlášený uživatel zadá název datasetu a vybere soubor ve formátu xlsx. Po stisknutí tlačítka uložení je uživateli vytvořen dataset, se kterým může později pracovat.

4. UseCase4: Editace datasetu. Uživateli je zpřístupněna nabídka všech jeho existujících datasetů. Uživatel si vybere jeden z těchto datasetů, který chce editovat. Součástí editace je možnost změnit název datasetu a pomocí zobrazené tabulky a dvou výběrových seznamů, z nichž jeden obsahuje všechny možné atributy (sloupce tabulky) a druhý uživatelem vybrané atributy (sloupce tabulky), uložit do systému požadovaná data (vybrané atributy/sloupce tabulky).

5. UseCase5: Práce s reportovacím nástrojem. Tento scénář je nejrozsáhlejším scénářem, neboť tvoří hlavní funkci vyvíjeného systému. Po otevření reportovací části aplikace se uživateli nabízí možnost tvořit vlastní reporty, které se skládají z grafů, textů, nadpisů a tabulek. Tato

část se skládá ze seznamu položek, který obsahuje přidané grafy, tabulky, texty a nadpisy, jenž uživatel může do tohoto seznamu vkládat a editovat je. Druhá část představuje náhled, jak bude výsledný pdf dokument vypadat. Poté má uživatel možnost ukládat definované grafy a tabulky s konkrétními daty. Uživatel má možnost report uložit ve formátu pdf a nebo také do systému, ze kterého tento report může později načíst.

3.2.1 První případ užití - UseCase1: Příhlášení

Aktéři

- Uživatel
- Systém

Podmínky spuštění

Uživatel se musí nacházet v částí aplikace, která obsahuje přihlašovací formulář.

Základní tok

1. Systém zobrazí formulář, který obsahuje vstup pro zadání emailu a vstup pro zadání hesla.
2. Uživatel vyplní email.
3. Uživatel vyplní heslo.
4. Uživatel odešle formulář příslušným tlačítkem submit.
5. Systém přihlásí uživatele v případě, že byla správně zadána kombinace jména a hesla.
6. Scénář končí.

Alternativní tok

5.1 Systém zobrazí chybovou hlášku v případě, že uživatel nebyl nalezen jako registrovaný uživatel v databázi nebo kombinace jména a hesla není správná.
5.2 Tok se vrací zpátky ke kroku 2.

3.2.2 Druhý případ užití - UseCase2: Registrace účtu

Aktéři

- Uživatel
- Systém

Podmínky spuštění

Uživatel se musí nacházet v části aplikace, která obsahuje registrační formulář.

Základní tok

1. Systém zobrazí formulář, který vyzve uživatele k zadání jména, emailu a hesla.
2. Uživatel vyplní jméno.
3. Uživatel vyplní email.
4. Uživatel vyplní heslo.
5. Uživatel odešle formulář tlačítkem submit.
6. Systém registruje nového uživatele, který může použít své nové přístupové údaje k přihlášení.
7. Systém automaticky přihlásí nově registrovaného uživatele.
8. Scénář končí úspěšnou registrací uživatele.

Alternativní tok 1

2.1 Systém zobrazí chybovou hlášku, pokud jméno nebylo vyplněno.
2.2 Tok se vrací zpátky ke kroku 2.

Alternativní tok 2

3.1 Systém zobrazí chybovou hlášku, pokud email není ve validním formátu (jmeno@server.domena).
3.2 Tok se vrací zpátky ke kroku 3.

Alternativní tok 3

4.1 Systém zobrazí chybovou hlášku v případě, že heslo je kratší než 5 znaků.
4.2 Tok se vrací ke kroku 4.

3.2.3 Třetí případ užití - UseCase3: Přidání datasetu

Aktéři

- Uživatel
- Systém

Podmínky spuštění

Uživatel se nachází v části aplikace, která obsahuje formulář pro vložení nového datasetu. Dále uživatel musí být přihlášen.

Základní tok

1. Systém zobrazí formulář, v němž jsou obsaženy vstupy pro zadání názvu i popisku datasetu a samotný vstup pro nahrání souboru ve formátu xlsx nebo xls.
2. Uživatel vyplní název vytvářeného datasetu. Název datasetu je povinná položka.
3. Uživatel vyplní popisek k datasetu. Popisek musí obsahovat alespoň 5 znaků. Slouží pro bližší specifikaci, jaká data daný dataset obsahuje.
4. Uživatel klikne na příslušné tlačítko pro výběr souboru.
5. Systém zobrazí uživateli dialogové okno pro výběr souboru.
6. Uživatel ve svém lokálním systému vybere požadovaný xlsx nebo xls soubor.
7. Systém zaznamená načtení daného souboru.
8. Uživatel potvrdí vložení nového datasetu pomocí tlačítka.
9. Scénář končí.

Alternativní tok 1

2.1 Systém zobrazí chybovou hlášku právě tehdy, když název datasetu není vyplněn.
2.2 Tok se vrací ke kroku 2.

Alternativní tok 2

3.1 Systém zobrazí chybovou hlášku v případě, kdy popisek je kratší než 5 znaků.
3.2 Tok se vrací ke kroku 3.

3.2.4 Čtvrtý případ užití - UseCase4: Editace datasetu

Aktéři

- Uživatel
- Systém

Podmínky spuštění

Uživatel je přihlášen. Uživatel se nachází v části aplikace, jež zobrazuje seznam existujících datasetů daného uživatele.

Základní tok

1. Systém zobrazí seznam existujících datasetů uživatele.
2. Uživatel zvolí možnost editace vybraného datasetu pomocí tlačítka pro editaci.
3. Systém zobrazí uživateli formulář, který obsahuje vstup pro název datasetu, tabulku a dva seznamy, mezí kterými lze přesouvat data.
4. Uživatel může přepsat existující název datasetu anebo jej ponechat.
5. Uživatel si zvolí pomocí dvou výběrových seznamů, která data budou použita v daném datasetu. První seznam obsahuje veškeré atributy (sloupce tabulky), které nejsou použity v datasetu a druhý obsahuje veškeré atributy (sloupce tabulky), které jsou použity v datasetu.
6. Uživatel zvolí možnost uložení změn.
7. Systém přepíše existující dataset.
8. Scénář končí.

Alternativní tok 1

2.1 Uživatel zvolí možnost smazat daný dataset.
2.2 Systém zobrazí modální okno, kde je uživateli zobrazena otázka, zda-li chce skutečně daný dataset smazat.
2.3 Uživatel potvrdí možnost smazání.
2.4 Systém smaže daný dataset.
2.5 Systém zavře modální okno a zobrazí uživateli seznam datasetů.

Alternativní tok 2

2.2.1 Uživatel zvolí možnost zrušit mazání.
2.2.2 Systém zavře modální okno pro mazání.
2.2.3 Systém zobrazí uživateli seznam datasetů.

Alternativní tok 3

6.1 Uživatel se může přepnout do jiné části aplikace.
6.2 Systém zruší editaci datasetu a neprovede žádné změny.
6.3 Systém navede uživatele na požadovanou část aplikace.

3.2.5 Pátý případ užití - UseCase5: Práce s reportovacím nástrojem

Aktéři

- Uživatel
- Systém

Podmínky spuštění

Uživatel je přihlášen. Uživatel se nachází v části aplikace, která slouží pro práci s reporty.

Základní tok

1. Systém zobrazí uživateli formulář pro práci s reporty.
2. Systém zobrazí první část formuláře, kdem ohou být vkládány uživatelem jednotlivé položky reportu. Tyto položky mohou být: nadpis, graf, text a tabulka.
3. Systém zobrazí druhou část, která zobrazuje náhled výsledného reportu.
4. Systém zobrazí tlačítko pro přidání nového grafu.
5. Systém zobrazí tlačítko pro přidání nové tabulky.
6. Systém zobrazí tlačítko pro přidání nového textu.
7. Systém zobrazí tlačítko pro přidání nového nadpisu.
8. Systém zobrazí tlačítko pro možnost načtení existujícího reportu z databáze.
9. Systém zobrazí tlačítko pro přidání existujícího grafu.
10. Systém zobrazí tlačítko pro přidání existující tabulky.
11. Systém zobrazí tlačítko pro načtení existujícího reportu.
12. Systém zobrazí tlačítko pro uložení definice reportu.
13. Systém zobrazí tlačítko pro vyrenderování vytvořeného reportu do druhé části sloužící pro náhled reportu.

14. Systém zobrazí u všech existujících položek reportu tlačítko pro možnost smazání ze seznamu položek reportu. Tuto akci systém provede pokaždé při nově vytvořené položce reportu.
15. Systém zobrazí tlačítko pro možnost stažení reportu ve formátu PDF.
16. Uživatel zvolí jednu z požadovaných akcí.
17. Scénář pokračuje krokem 15 nebo krokem 17.
18. Scénář končí.

Alternativní tok 1 - Přidání nového grafu

15.1.1 Uživatel vybere přidání nového grafu.
15.1.2 Systém vloží do první části pro položky reportu položku grafu.
15.1.3 Uživatel navolí požadované parametry grafu. Volba názvu grafu, datasetu, typu grafu a dodatečné parametry v závislosti na typu grafu.
15.1.4 Scénář pokračuje krokem 15 nebo 17.

Alternativní tok 2 - Přidání nové tabulky

15.2.1 Uživatel vybere možnost přidání nové tabulky.
15.2.2 Systém vloží do první části pro položky reportu položku tabulky.
15.2.3 Uživatel zvolí název tabulky.
15.2.4 Uživatel zvolí dataset, ze kterého se vytvoří tabulka.
15.2.5 Uživatel zvolí možnost zda-li shlukovat tabulku dle nějakého atributu.
15.2.6 Pokud je zvolena možnost shlukování, uživatel si vybere, kterou tabulku daného shluku chce vykreslit, případně může zvolit možnost vykreslit tabulky všech shluků.
15.2.7 Scénář pokračuje krokem 15 nebo 17.

Alternativní tok 3 - Přidání nového textu

15.3.1 Uživatel vybere možnost přidat nový text.
15.3.2 Uživatel vyplní obsah textového pole.
15.3.3 Scénář pokračuje krokem 15 nebo 17.

Alternativní tok 4 - Přidání nového nadpisu

15.4.1 Uživatel vybere možnost přidat nový nadpis.
15.4.2 Uživatel vyplní obsah textového pole pro nadpis.

15.4.3 Uživatel vybere velikost nadpisu pomocí výběrového seznamu.
15.4.4 Scénář pokračuje krokem 15 nebo 17.

Alternativní tok 5 - Přidání existujícího grafu

15.5.1 Uživatel vybere možnost přidat již existující graf.
15.5.2 Systém zobrazí modální okno s výběrovým seznamem existujících grafů.
15.5.3 Uživatel v modálním okně zvolí z výběrového seznamu existující graf.
15.5.4 Systém vloží do první části pro položky seznamu novou položku existujícího grafu.
15.5.5 Scénář pokračuje krokem 15 nebo 17.

Alternativní tok 6 - Přidání existující tabulky

15.6.1 Uživatel vybere možnost přidat již existující tabulku.
15.6.2 Systém zobrazí modální okno s výběrovým seznamem existujících tabulek.
15.6.3 Uživatel v modálním okně zvolí z výběrového seznamu existující tabulku.
15.6.4 Systém vloží do první části pro položky seznamu novou položku existující tabulky.
15.6.5 Scénář pokračuje krokem 15 nebo 17.

Alternativní tok 7 - Načtení reportu

15.7.1 Uživatel vybere možnost načtení reportu.
15.7.2 Systém zobrazí modální okno, ve kterém uživatel vybere jeden ze svých již existujících reportů.
15.7.3 Systém načte položky reportu do seznamu položek reportu.
15.7.4 Report je načten.
15.7.5 Scénář pokračuje krokem 15 nebo 17.

Alternativní tok 8 - Uložení reportu

15.8.1 Uživatel vybere možnost uložit report.
15.8.2 Systém načte ze vstupu název reportu.
15.8.3 Systém uloží položky seznamu reportu jako nový report.
15.8.4 Scénář pokračuje krokem 15 nebo 17.

Alternativní tok 9 - Renderování reportu

15.9.1 Uživatel vybere možnost renderování reportu.
15.9.2 Systém načte veškerá data ze seznamu položek reportu a vyrenderuje náhled reportu do druhé částí aplikace pro zobrazení náhledu reportu.
15.9.3 Scénář pokračuje krokem 15 nebo 17.

Alternativní tok 10 - Stažení reportu ve formátu PDF

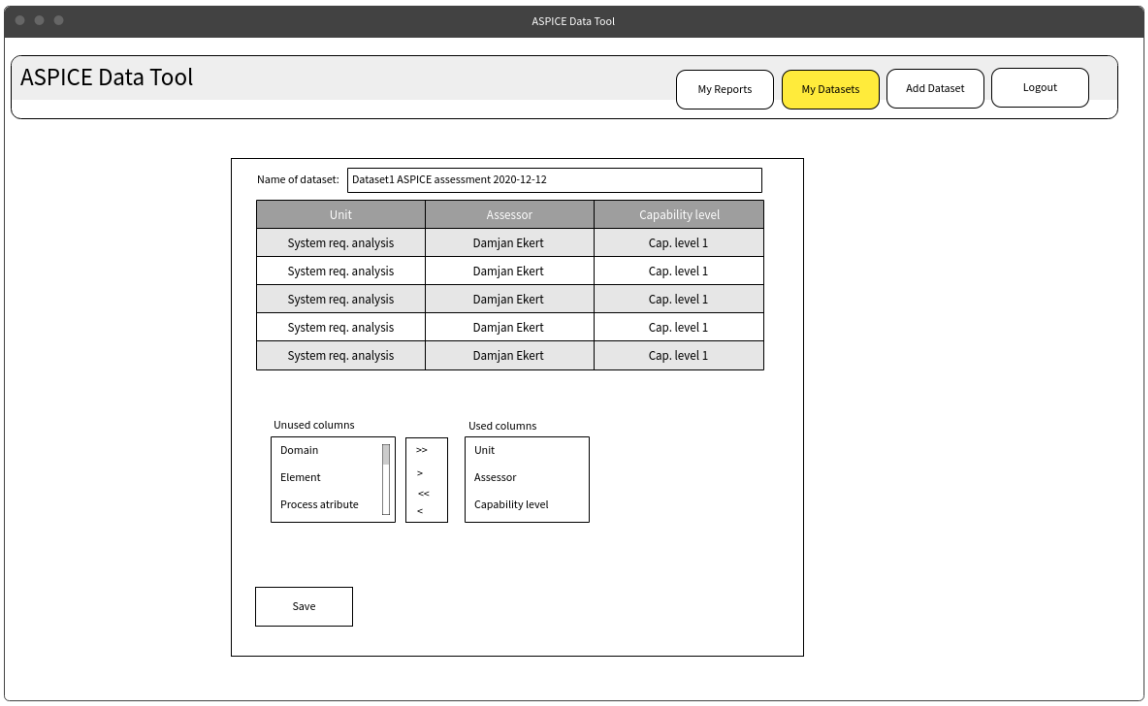
15.10.1 Uživatel vybere možnost stažení reportu.
15.10.2 Systém vyrenderuje report do formátu PDF a stáhne jej uživateli jako soubor v tomto formátu.
15.10.3 Scénář pokračuje krokem 15 nebo 17.

3.3 Uživatelské rozhraní

Uživatelské rozhraní bylo navrženo se snahou zaměřit se na přehlednost a komfort při používání. Aplikace se skládá z hlavního panelu, kde je zobrazen název aplikace a uživatelské menu. Pod uživatelským menu je část, kde se vykreslují požadované funkcionality. Dále systém obsahuje vlastní definovaná modální okna, která se zobrazí vždy pokud uživatel provede specifickou interakci s aplikací. Nakonec jednotlivé vizuální prvky aplikace jsou závislé na konkrétním scénáři, který právě uživatel vykonává. Výsledné vizuální zobrazení naimplementovaného systému se může v některých případech lišit.

3.3.1 Skica 1. - Návrh uživatelského rozhraní pro editaci datasetu

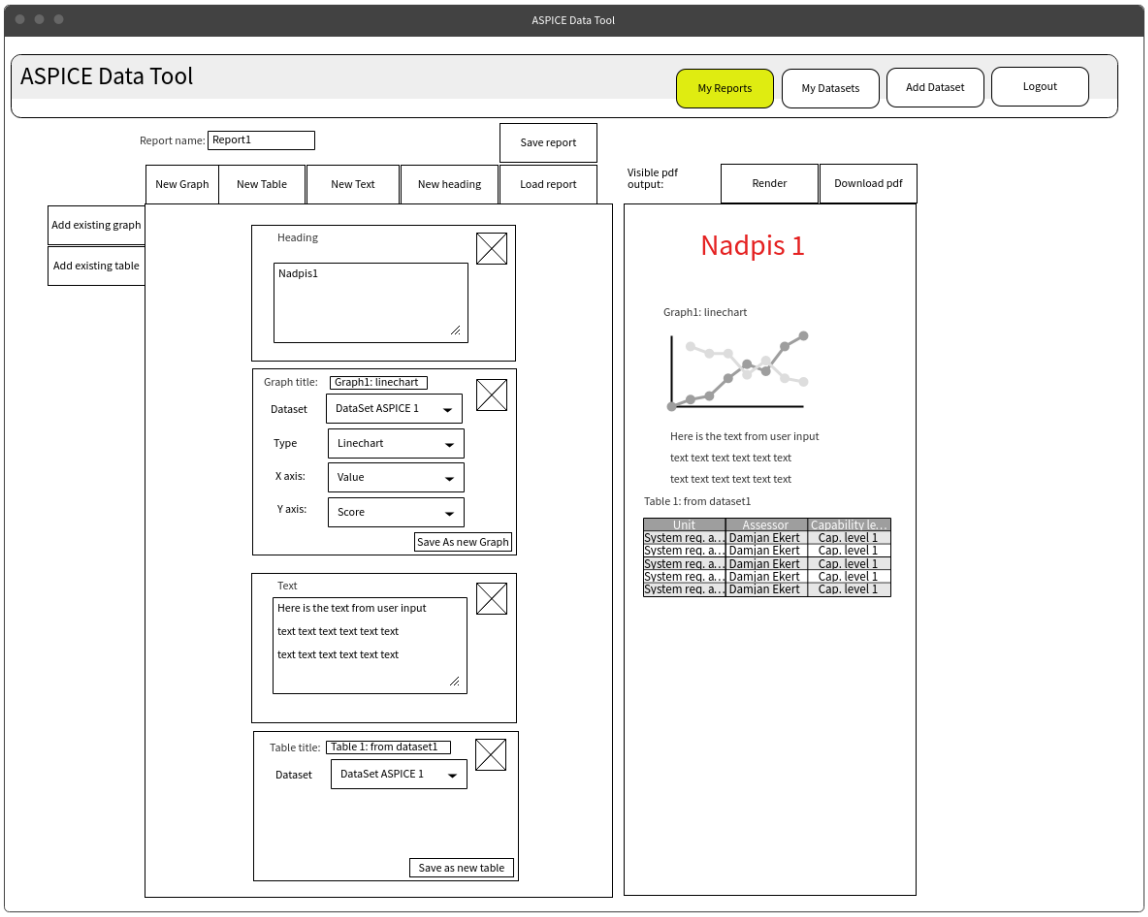
Na obrázku 3.2 je zobrazen návrh formuláře pro editaci datasetu. Skica rozhraní se skládá jak ze vstupního pole pro vložení resp. editaci názvu datasetu, tak z tabulky, která zobrazuje zvolená data na základě dvou seznamů, mezi kterými uživatel přesouvá dostupné a použité atributy a nakonec také z tlačítka pro uložení. Skica byla nakreslena pomocí webového programu MockFlow [14].



Obrázek 3.2: Návrh uživatelského rozhraní - editace datasetu.

3.3.2 Skica 2. - Návrh uživatelského rozhraní pro práci s reportovacím nástrojem

Obrázek 3.3 prezentuje formuláře pro vytváření a práci s reporty. Skica rozhraní se skládá z několika dílčích částí. V levém horním rohu je zobrazen název reportu. Pod názvem reportu se nachází sada tlačítek pro vytvoření nového grafu, nové tabulky, nového textu, nového nadpisu, načtení a uložení reportu. Pod těmito tlačítky je zobrazen seznam objektů položek reportu. Vlevo od tohoto seznamu se nachází tlačítko pro vložení existujícího grafu a tlačítko pro vložení existující tabulky. Vpravo od seznamu položek reportu se nachází okno, ve kterém je vykreslován náhled reportu po stisknutí tlačítka render, které se nachází nad tímto oknem. Vedle tlačítka render je tlačítko pro stažení reportu. Samotný seznam objektů reportu ve skice (obrázek 3.3) obsahuje ve všech svých položkách tlačítko pro odebrání ze seznamu a další prvky uživatelského rozhraní v závislosti na typu objektu položky reportu.



Obrázek 3.3: Návrh uživatelského rozhraní - práce s reportovacím nástrojem

Kapitola 4

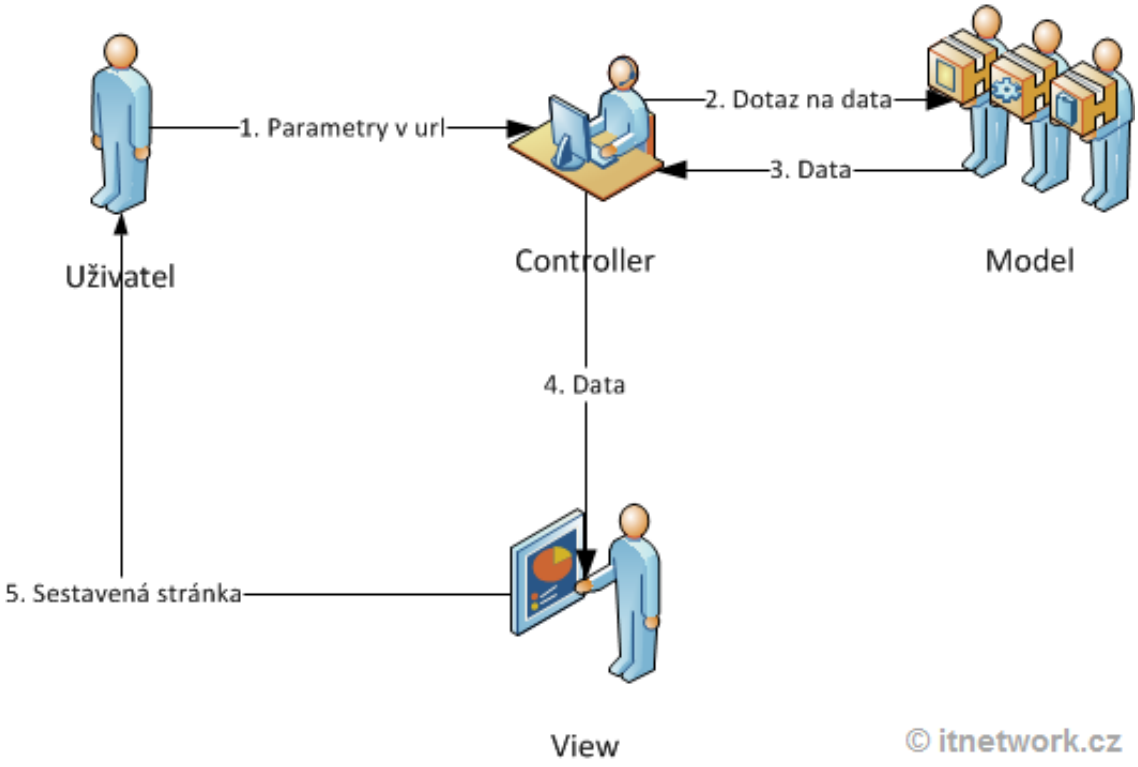
Architektura

V této kapitole je popsána architektura vyvíjeného systému. Architektura navrhovaného systému je jednou z klíčových otázek z hlediska strukturování aplikace a fungování jednotlivých částí jako celku. Popsáno je logické rozčlenění částí systému, způsob uchování dat a v krátkosti zvolené technologie.

4.1 Rozložení systému

V rámci návrhu systému byla zvolena architektura klient/server[15], která byla nejvhodnější volbou vzhledem k požadavkům na vyvíjený systém, jenž mimo jiné zahrnuje víceuživatelský přístup. Příkladem jiné architektury by mohla být peer to peer síť[16], která ovšem neobsahuje žádný centrální prvek, do kterého by se ukládala uživatelská data. Jediná možnost by tedy byla do takové sítě zavést distribuovanou databázi, ale pro tento typ úlohy by bylo toto řešení nevhodné. Nepsaným standardem dnešní doby je vyvíjet systémy jako webovou aplikaci. Tudíž klientská část vznikající aplikace bude právě webová. Serverová část bude sloužit jako mezivrstva mezi perzistentním úložištěm a stranou klienta. V rámci již zmíněné klient/server architektury pro vznikající systém, byl zvolen architektonický vzor MVC[17] neboli Model View Controller (obrázek 4.1). V implementační části práce budou model, view i controller popsány detailněji.

Hlavní část aplikace tvoří rozhraní pro tvorbu reportů z daného datasetu. Z tohoto důvodu byla zvolena strategie takzvaného tlustého klienta. Volba tlustého klienta přispívá k celkové responzivitě vyvíjeného systému, kdy většina výpočtů, stavů a algoritmů je součástí klientské strany. Uživatel tak nemusí ve většině případů čekat na odpověď serveru, který by v rámci volby tenkého klienta obsahoval business logiku. Další výhodou zvolení tlustého klienta je omezení celkového síťového provozu, což může obecně vést k lepším odezvám serverové strany, která v jeden okamžik zpracovává požadavky od více uživatelů. Klient si tak v daný okamžik načte ze serveru všechna potřebná data, se kterými uživatel bude eventuelně pracovat a poté se doptává serveru na dodatečná data, pokud uživatel změní stav aplikace.

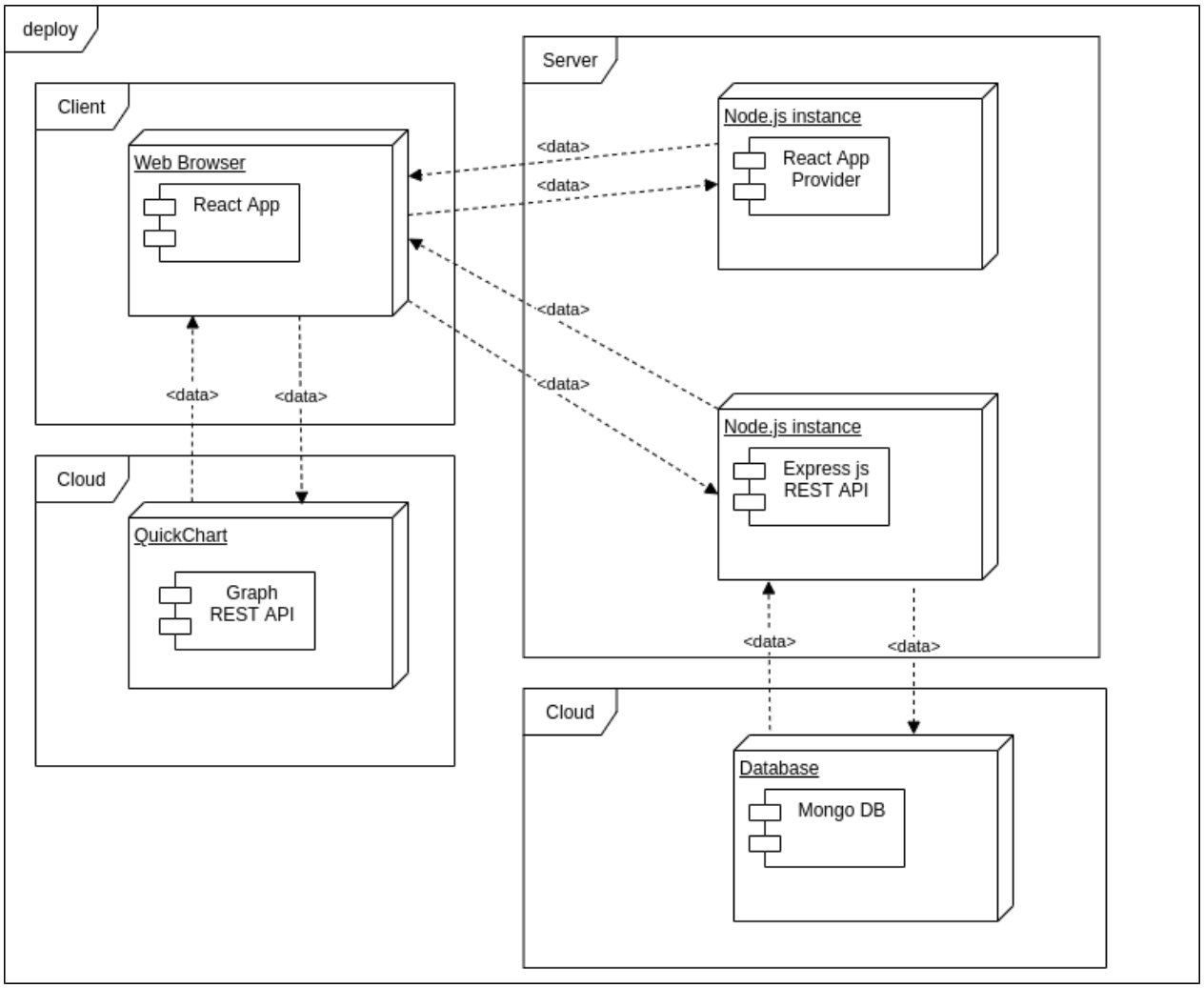


Obrázek 4.1: Diagram Model View Controller [17]

Volba tlustého klienta automaticky vede k použití serverové části jako mezivrstvy mezi úložištěm a klientem, jak již bylo zmíněno na začátku.[18] Nabízí se dvě možnosti komunikačních protokolů webových služeb, které by předávaly data mezi klientem a serverem. První možností je volba protokolu SOAP (Simple Object Access Protocol).[19] SOAP přenáší data mezi klientem a webovou službou (serverem) ve formátu XML (Extensible Markup Language).[20] Výhodou volby tohoto protokolu by byla možnost vestavěné logiky při selhání komunikace. Druhou možností je protokol REST (Representational State Transfer), který byl upřednostněn před protokolem SOAP, díky podpoře datového formátu JSON. Hlavní výhodou oproti protokolu SOAP je možnost volby formátu dat. REST například umožňuje předávat data ve formátu JSON, PDF, SVG a nebo stejně jako předchozí zmíněný protokol ve formátu XML.[20] V rámci protokolu REST klient nepracuje přímo se zdrojem dat, ale s jeho reprezentací. Systém bude odesílat z klientské strany na server soubory formátu xlsx a data obousměrně ve formátu JSON.[20] Výhodou formátu JSON je rychlejší parsování a konvertování na objekty. Nevýhodou formátu JSON jsou binární data, se kterými tento formát nenakládá optimálně.

4.2 Návrh a Technologie

Volba technologií plynule navazuje na předchozí část, kde byl zvolen architektonický styl a strategie. Pro klientskou část byla zvolena javascriptová knihovna React[21], která má početnou komunitu a je neustále vyvíjena. Serverová část využívá webový framework Express[22], který je postaven také na skriptovacím jazyce javascript. Provider klienta i server běží ve své vlastní instanci technologie běhového prostředí pro javascript[23] Node.js.[24] Volba takzvané NoSQL[25] databáze MongoDB[26] byla založena na volbě jazyka a frameworku serverové části. Databáze MongoDB se k výše zmíněným technologiím, vzhledem ke své jednoduchosti a požadavkům na aplikaci, hodí. Poslední technologií je QuickChart[27], která slouží ke generování grafů ve formátu obrázku pomocí volání REST API[28] na veřejně dostupnou službu, která je umístěna v cloudu. Diagram nasazení zmíněných technologií je zobrazen na obrázku 4.2.

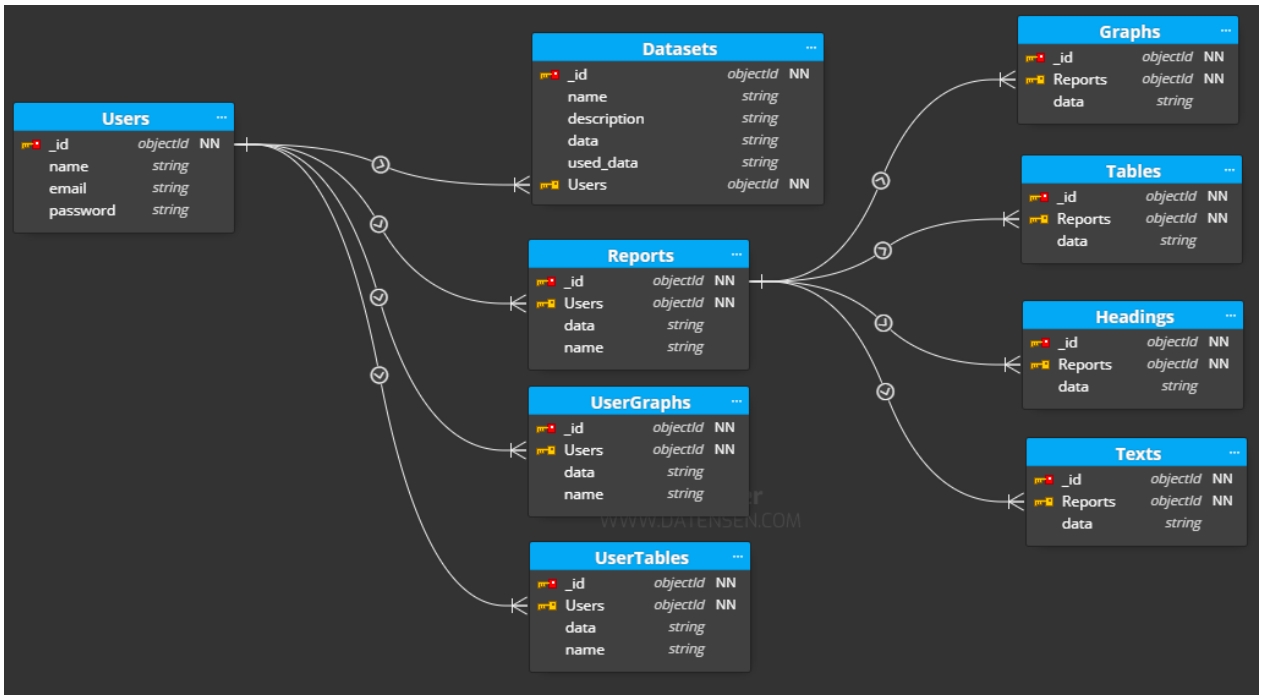


Obrázek 4.2: Diagram nasazení

4.2.1 Databáze

Vzhledem k variabilitě dat, která se mohou na vstupu vyskytovat, byla zvolena databáze MongoDB. Databáze MongoDB se řadí mezi takzvané NoSQL databáze[25], které obecně nemají typické vlastnosti relačních databází. Data nejsou uchovávána v klasických n-ticích, které jsou ve všech případech použity u relační databáze. Databázový systém MongoDB drží databáze, které obsahují kolekce a tyto kolekce obsahují dokumenty. Dokumentem je v tomto případě chápán dokument typu JSON. Každý záznam je ukládán právě jako dokument ve formátu JSON. Kolekce shromažďují dokumenty stejného typu. Absence dotazovacího jazyka SQL v této databázi je nahrazena knihovnou mongoose[29], která umožňuje vytvořit modely, nad nimiž lze provádět vybrané funkce, které jsou schopny spolehlivě SQL dotazy nahradit.

Výhodou této NoSQL databáze je možnost ukládat různorodá data.[25] Z této vlastnosti těží celý navržený systém, který je schopen pracovat s jakýmkoliv datasetem, nezávisle na názvech sloupců/atributů. Nevýhoda MongoDB oproti relačním databázím taková, že ve výchozím stavu nejsou zaručeny typické vlastnosti relačních databází ACID[30]. Nicméně MongoDB má mechanismy, pomocí kterých lze implementovat transakce a například integritní omezení. Relace jsou řešeny pomocí odkazování se na unikátní ID odkazované entity nebo vložením celého serializovaného objektu do konkrétního atributu. Navržené entity, které jsou v rámci systému využívány, jsou zobrazeny na obrázku 4.3.



Obrázek 4.3: Schéma jednotlivých entit a vztahů mezi nimi

Popis jednotlivých entit:

- **Users** - Entita, která drží informace o uživatelech systému. Každý uživatel má jméno, heslo a email. Pro ověřování používá kombinaci emailu a hesla.
- **Datasets** - Entita, která drží informace o jednotlivých datasetech daného uživatele. Každý dataset byl do systému nahrán jako xlsx soubor, který byl převeden do JSON formátu a uložen do atributu data. Dataset má své jméno, popisek a odkaz na uživatele, kterému dataset patří.
- **Reports** - Entita, která drží informace o uživatelem definovaném reportu. Obsahuje název a odkaz na uživatele, kterému daný report patří. V atributu data jsou pak uloženy objekty, ze kterých se report skládá (tabulky, grafy, nadpisy a texty).
- **UserGraphs** - Entita, která ukládá informace o grafech, které si nadefinoval uživatel. Tento graf může být vložen do reportu. Skládá se z názvu, odkazu na daného uživatele a z atributu data, který obsahuje veškeré data spojená s objektem grafu.
- **UserTables** - Entita, která ukládá informace o tabulkách, které definoval uživatel. Existující tabulku lze vložit do reportu. Skládá se z atributu pro název, odkaz na uživatele a data. Atribut pro data obsahuje data, která jsou s objektem tabulky spjata.
- **Graphs** - Entita, která je součástí dat (atribut data) entity Reports. Obsahuje veškeré informace o konkrétním grafu, které drží objekt grafu.
- **Tables** - Entita, která je také součástí dat (atribut data) entity Reports. Obsahuje veškeré informace o konkrétní tabulce, které drží objekt tabulky.
- **Headings** - Entita, která je součástí dat (atribut data) entity Reports. Obsahuje veškeré informace o konkrétním nadpisu, které drží objekt nadpisu.
- **Texts** - Entita, která je součástí dat (atribut data) entity Reports. Obsahuje veškeré informace o konkrétním textu, které drží objekt textu.

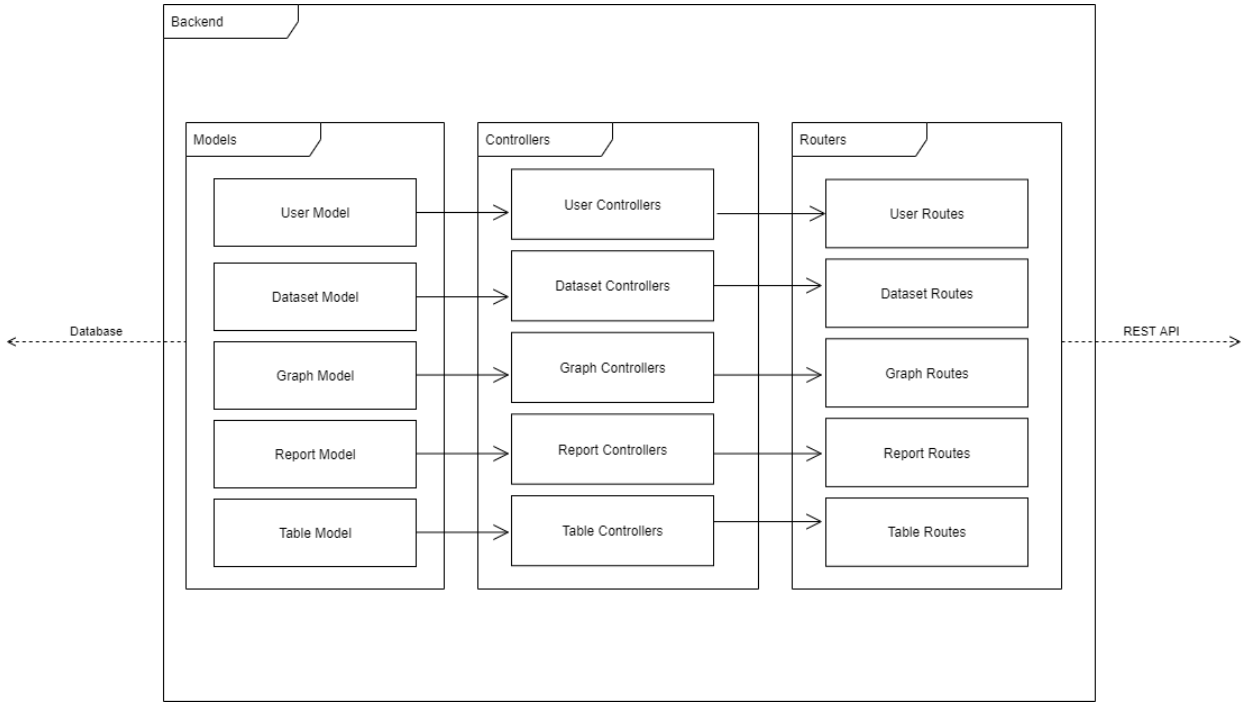
4.2.2 Server

Serverová část systému se skládá ze dvou dílčích aplikaci. V první javascriptové aplikací je využit webový framework Express, který v implementovaném systému slouží jako middleware[31] pro zprostředkování dat. Express[22] zastřešuje http komunikaci mezi klientem a serverem. Umožňuje pomocí takzvaného volání middleware funkcí obsluhovat požadavky. Framework běží v běhovém prostředí javascriptu Node.js[24]. Druhou aplikací na serveru je aplikace psaná v Reactu, která zpřístupňuje uživatelské rozhraní pro uživatele a také běží v běhovém prostředí Node.js. Každá z těchto aplikací běží na samostatném aplikačním portu. Samotní uživatelé se však připojují z webového prohlížeče

pouze na aplikaci uživatelského rozhraní. Bližší specifikace této aplikace klienta budou upřesněny v pozdější kapitole návrhu Klienta.

Následující popis bude tedy věnován samotné backendové části celého systému. Využití frameworku Express se neobejde bez návrhu, který určí, jakým způsobem bude server poskytovat navenek data. Jak již bylo popsáno, server bude poskytovat data pomocí protokolu REST. Je tedy nutné navrhnout a poté naimplementovat aplikační rozhraní pro tento protokol. Rozhraní protokolu REST bude realizováno pomocí routerů, které specifikují cestu k požadovaným operacím na datech. Tyto routery budou poté volat metody konkrétních controllerů, které zajišťují požadované operace. Controller je termín z modelu MVC a má za cíl manipulovat s daty, které pomocí volání funkcí nad modely z knihovny mongoose načítá a ukládá do databáze.

Controller v tomto případě však nezajišťuje změny přímo ve View, ale ve spojení s routery poskytuje pomocí REST data klientské aplikaci, která celkově slouží jako View. Na obrázku 4.4 je zobrazeno schéma závislostí jednotlivých komponent backendové části. Modely poskytují vrstvu pro komunikaci s databází a provádějí databázové operace. Zároveň slouží jako reprezentanti dat. Controllery využívají modely, nad kterými volají specifické databázové operace a načítají, případně ukládají do modelů objekty. Routery poskytují navenek cesty, které jak již bylo řečeno, slouží jako rozhraní protokolu REST. Pro každou cestu existuje metoda konkrétního controlleru, která má na starost provést požadovanou operaci.



Obrázek 4.4: Diagram závislostí komponent backendové části systému

Navržené aplikační rozhraní se skládá z následujících cest:

User

- "/api/users/"- při volání této cesty pomocí http metody GET vrátí server data ve formátu JSON. Vrácená data obsahují pole všech objektů typu User. V případě neúspěchu vrací objekt s chybovou zprávou.

- "api/users/signup"- pomocí http metody POST a vložením dat ve formátu JSON do tohoto http požadavku se vytvoří nový uživatel. Vstupní data musí obsahovat hodnoty pro name, email a password (jméno, email a heslo). Jméno nesmí být prázdné, email musí mít validní formát a heslo musí být minimální délky 7. V případě neúspěchu se vrací objekt s chybovou zprávou.

- "api/users/login"- kombinací http metody POST a vložením dat do http požadavku ve formátu JSON server vrátí objekt, který obsahuje informace o tom, zda byl uživatel úspěšně přihlášen či nikoli. Přihlášení je úspěšné, pokud je kombinace emailu a hesla v porovnání s daty uloženými v databázi shodná. Vstupní data tvoří email a heslo.

Dataset

- "/api/datasets/newdataset"- Http metoda POST a vstupní data požadavku ve formátu JSON, která obsahují name, description, user ID a file (jméno datasetu, popisek datasetu a ID uživatele, kterému patří dataset a soubor) zajistí vložení nového datasetu do databáze. Soubor je typu xlsx. V případě úspěchu server vrací vytvořený objekt serializovaný do formátu JSON.

- "api/datasets/getusersdatasets"- Pomocí http metody POST a vstupních JSON dat požadavku lze od serveru obdržet pole objektů typu dataset určitého uživatele. Vstupní data obsahují ID uživatele, pro kterého se mají dané datasety vyhledat a vrátit. Při selhání se vrací chybová zpráva.

- "api/datasets/getdataset"- Metoda POST protokolu http a požadavek obsahující JSON data s unikátním ID existujícího datasetu jsou zaslána na server, zajistí vrácení objektu typu dataset dle definovaného ID.

- "api/datasets/editdataset"- Pomocí http metody PATCH a vstupu formátu JSON daného požadavku server příjme tato data a upraví existující dataset. Vstupem pro editování datasetu je name, description, dataset ID a used data (jméno, popis, ID datasetu, který se má editovat a data, která jsou uživatelem vybrána k použití v aplikaci). V případě úspěchu server vrací zprávu o úspěšném editování zvoleného datasetu. V opačném případě vrátí zprávu o neúspěchu.

- "api/datasets/deletedataset"- Http metoda DELETE a vstupní data obsahující ID existujícího datasetu zajistí smazání záznamu konkrétního datasetu. V případě úspěchu server vrátí zprávu o úspěšném smazání.

Report

- "/api/reports/reports"- Pomocí http metody POST a vstupních dat požadavku obsahující ID uživatele, vrací server jako odpověď v případě úspěchu pole objektů typu report ve formátu JSON.
- "/api/reports/new"- Metoda POST a vstupní data http požadavku obsahující name, data a user ID (název, data obsahující veškeré objekty reportu a ID uživatele, pro kterého bude vytvořen report) odešle tato data na server, který požadavek zpracuje a v úspěšném případě vytvoří nový záznam typu report.
- "/api/reports/get"- Vstupní data http požadavku, která obsahují id reportu a http metoda POST získá ze serveru objekt typu report v JSON formátu. Pokud server žádný takový záznam nenalezne, vrátí zprávu o neúspěchu při vyhledávání.

Graph

- "/api/graphs/graphs"- Pomocí http metody POST a vstupních dat požadavku obsahující ID uživatele, vrací server jako odpověď v případě úspěchu pole objektů typu graf ve formátu JSON.
- "/api/graphs/new"- Metoda POST a vstupní data http požadavku obsahující name, data a user ID (název, data obsahující veškeré atributy objektu grafu a ID uživatele, pro kterého bude vytvořen graf) odešle tato data na server, který požadavek zpracuje a v úspěšném případě vytvoří nový záznam typu graf.
- "/api/graphs/get"- Vstupní data http požadavku, která obsahují ID grafu a http metoda POST získá ze serveru objekt typu graf v datovém formátu JSON. Pokud server žádný takový záznam nenalezne, vrátí zprávu informující o neúspěchu při vyhledávání.

Table

- "/api/tables/tables"- Pomocí http metody POST a vstupních dat požadavku obsahující ID uživatele, vrací server jako odpověď v případě úspěchu pole objektů typu tabulka ve formátu JSON.
- "/api/tables/new"- Metoda POST a vstupní data http požadavku obsahující name, data a user ID (název, data obsahující veškeré atributy objektu tabulky a ID uživatele, pro kterého bude vytvořena tabulka) odešle tato data na server, který požadavek zpracuje a v úspěšném případě vytvoří nový záznam typu table.

- "/api/tables/get"- Vstupní data http požadavku, která obsahují ID tabulky a http metoda POST získá ze serveru objekt typu tabulka v datovém formátu JSON. Pokud server žádný takový záznam nenalezne, vrátí zprávu informující o neúspěchu při vyhledávání.

Pokud vznikne požadavek, který neodpovídá jedné z výše uvedených cest aplikačního rozhraní, pak server vrátí zprávu o nenalezení požadované cesty.

4.2.3 Klient

Klientská aplikace využívá knihovnu React. Aplikace se skládá z několika navržených dílčích částí. Tyto části jsou rozděleny obdobným způsobem, jakým knihovna Reactu nahlíží na svou strukturu. Každá navržená část se skládá z implementovaných funkčních a třídních komponent a stránek. Stránky jsou složeny z vlastních naimplementovaných komponent, nativních komponent a případně z komponent, které obsahuje nějaká využitá knihovna. Hlavní části tvoří sady komponent: datasets, reports, services shared a user.

Sada datasets má 3 stránky, které je možno zobrazit uživatelem v rámci webové aplikace. První ze stránek je stránka NewDataSet, která zpřístupní uživateli funkcionalitu pro vložení nového datasetu. Obsahuje formulář pro zadání jména, popisku a vstupního xlsx souboru.

Druhá stránka je UpdateDataSet, která zpřístupní uživateli možnost editovat svůj dataset. Editace datasetu spočívá ve vybrání sloupců (atributů), které mají být v reportovací části aplikace využity. Případně poskytuje možnost změny názvu datasetu.

Stránka UserDataSets zobrazí uživateli seznam všech jeho již existujících datasetů, které byly právě tímto uživatelem do systému nahrány. Zároveň se u všech zobrazených položek typu dataset zobrazí možnost editovat nebo smazat daný dataset. Stránka UserdataSets využívá komponentu DataSetList, která má za úkol zobrazit komponenty DataSetItem. Komponenta DataSetList mapuje datasety do jednotlivých DataSetItem komponent, které zobrazují informace o jednom datasetu uživatele a poskytují možnost tento dataset editovat nebo smazat.

Sada uživatele user se skládá z jedné stránky, která zpřístupňuje uživatelům systému přihlášení a registraci. Využívá sdílenou componentu Card, která zastřešuje design bloku pro přihlášení a registraci. Při přihlášení je využit autentifikační kontext, do kterého jsou v případě úspěšného přihlášení nahrána data o uživateli.

Reportovací část aplikace má sadu, která obsahuje několik komponent pro zajištění své funkcionality. Celá reportovací část se skládá z jedné stránky ReportPage, na které se podle stavu aplikace mění jednotlivé vizuální prvky aplikace, či se odesílají data na server, případně přijímají data ze serveru. Rovněž je součástí této stránky renderování požadovaných komponent do výsledného pdf reportu, který je poté možné stáhnout.

Stránka obsahuje komponentu Report, která zastřešuje všechna akční tlačítka pro vkládání nebo načítání prvků reportu a také obsahuje plátno, do kterého se jednotlivé prvky reportu při použití

akčních tlačítek vykreslují. Vložené prvky reportu mají pak podle svých vlastností nastavitelné položky.

Druhou a zároveň poslední komponentou, která je ve stránce ReportPage, je takzvaný PDF-Viewer. Ten má na starost zobrazit, po úspěšném vyrenderování uživatelem vložených prvků reportu, náhled pdf souboru, který může být v tomto náhledu různě přibližován. V náhledu může být zvolena možnost stažení souboru ve formátu pdf nebo přímého tisku dokumentu.

Komponenta Report, která jak již bylo zmíněno, je součástí stránky ReportPage, zajišťuje funkcionalitu veškerých akcí, které mohou být v reportovací části aplikace vykonávány. K tomu využívá komponenty, které slouží pro zobrazování jednotlivých tlačítek a prvků reportu. Tyto komponenty jsou následující:

- ReportButton - Tlačítko s předem nastavenými styly, které může být použito a napárováno s handlerem nějaké události během tvorby reportu. Styl tlačítka lze měnit pomocí vstupních vlastností.
- TextReportItem - Komponenta, která slouží pro zobrazení prvku reportu Text. Zajišťuje vložení uživatelem vepsaného textu do výsledného reportu.
- HeadingReportItem - Komponenta, která slouží pro zobrazení prvku reportu Heading (nadpis). Zastřešuje možnost vložení textu, který bude nadpis obsahovat a možnost volby jaké velikosti nadpis bude.
- TableReportItem - Jedná se o komponentu, jež má za úkol uživateli zobrazit ve výsledném reportu tabulku nebo sadu tabulek při vložení prvku reportu Table. Mezi vlastnosti patří výběr datasetu, který se má pro vykreslení tabulky použít, možnost zvolit group by dle nějakého sloupce tabulky (atributu datasetu) a poté zvolení možnosti vykreslení všech tabulek zvolených skupin, dle group by nebo nějaké jedné konkrétní tabulky z vytvořené skupiny pomocí group by.
- GraphReportItem - Komponenta, jež slouží k vložení uživatelem definovaného grafu do výsledného reportu. Umožní uživateli nastavit dataset, ze kterého se mají pro daný graf vzít data, poté umožní zvolit typ grafu, který může být buď sloupcový nebo koláčový, včetně možnosti zvolit group by, která zajišťuje vytvoření skupin dle nějakého atributu datasetu. Poté lze vybrat z datasetu atribut, nad kterým se provede agregační funkce sumace nebo zprůměrování v dané vybrané skupině. Pokud uživatel nezvolí možnost group by, pak uživatel zvolí popisky osy x a hodnoty osy y volbou atributů datasetu. Komponenta report se pak dále postará pomocí svých handlerů o odeslání potřebných dat a následné získání obrázku grafu z cloudové technologie QuickChart[27]. Ten je pak vložen do výsledného reportu při renderování.

Pro upřesnění, jakým způsobem bude uživatel a systém interagovat během případu tvorby reportu a práce s reportem, byly vytvořeny následující sekvenční diagramy. Diagramy na sebe navazují,

jelikož popisují stejnou část systému, neboť z důvodu rozsáhlosti popisu sekvenčním diagramem došlo k rozdělení jednoho sekvenčního diagramu do dvou diagramů. Diagramy rozšiřují náhled na funkčnost, která již byla popsána pomocí diagramu případů užití.

Diagram na obrázku 4.5 znázorňuje několik sekvencí, ke kterým v reportovací části aplikace může dojít. První ze sekvencí (alt1) je volitelná alternativní sekvence, kdy uživatel může libovolně měnit název reportu. Během přípravy reportovací části si webový klient načte data o datasetech pro daného přihlášeného uživatele. Tato data získá ze serveru pomocí volání REST API. Získaná data drží webový klient jako součást svého stavu. Data jsou později použita v položkách reportu.

Další ze sekvencí (alt2) je přidání nové položky do seznamu položek reportu (ReportItems). Uživatel má na výběr přidat buď text, nadpis, graf nebo tabulku. Webový klient si při této akci vytvoří objekt daného typu položky reportu, který si přidá do seznamu položek reportu, který je součástí stavu aplikace.

Sekvence alt3 popisuje způsob uložení tabulky nebo grafu daného názvu do databáze. Uživatel si později může takto uložené tabulky nebo grafy načíst. Webový klient nalezne objekt, který si uživatel přeje uložit, odešle jej pomocí REST API na server a server se postará o jeho uložení do databáze. Webový klient poté informuje uživatele o úspěšném uložení objektu.

Následující sekvence alt4 popisuje načtení položky existujícího grafu nebo tabulky, která byla uložena pomocí sekvence alt3. Webový klient odešle serveru požadavek pro získání existujícího objektu (grafu nebo tabulky), kterou si uživatel vybral pro načtení. Server vrátí webovému klientovi objekt grafu nebo tabulky. Takto načtený graf nebo tabulku si webový klient uloží do existujícího seznamu položek reportu, který se aktualizuje, aby uživatel viděl načtenou položku.

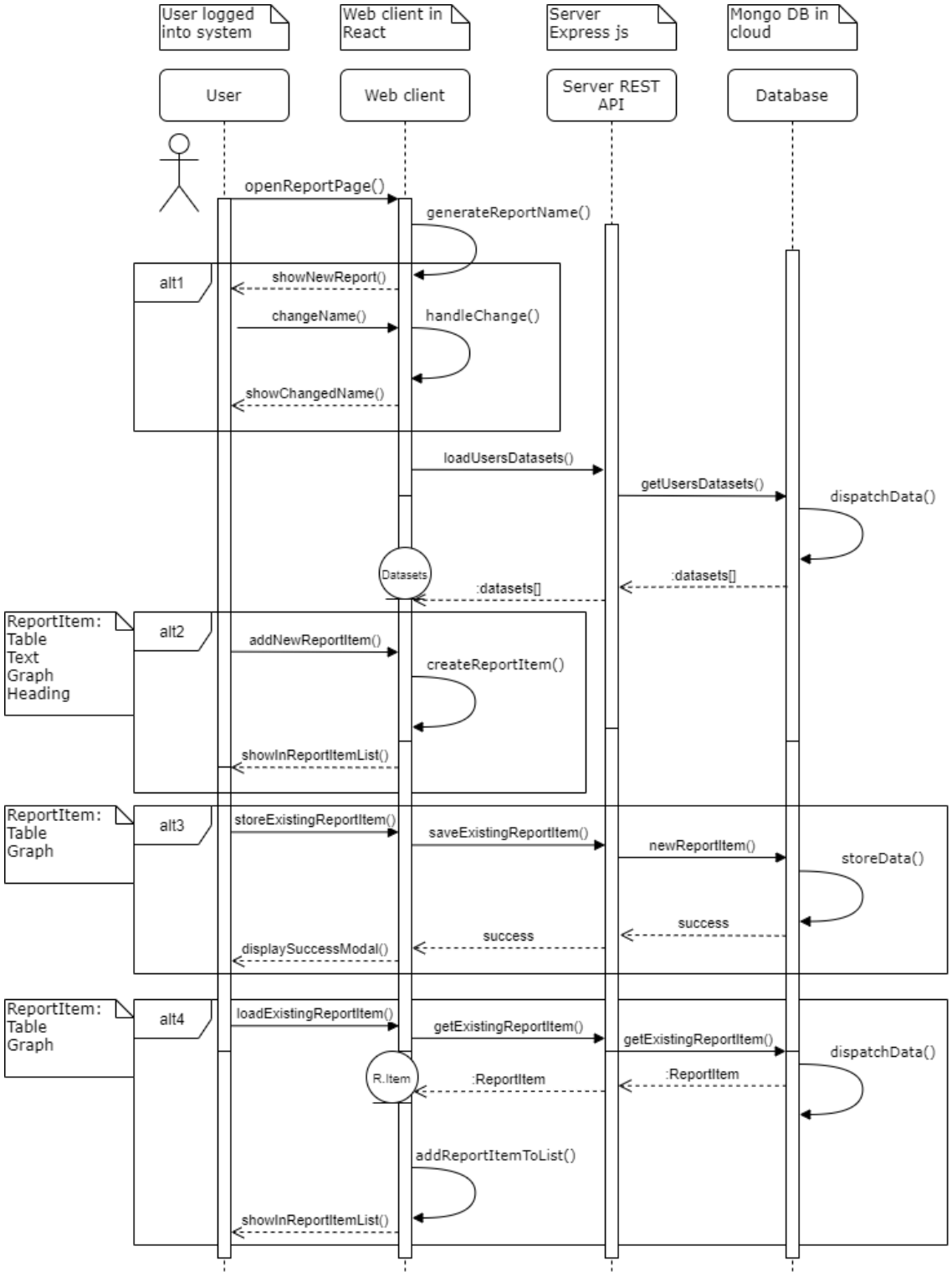
Diagram na obrázku 4.6 zobrazuje zbytek sekvencí, které nebyly obsaženy v prvním diagramu. Diagram plně navazuje na předešlý diagram znázorněný na obrázku 4.5.

Sekvence alt5 znázorňuje funkcionalitu načtení existujícího reportu z databáze. Webový klient odešle požadavek na server, který získá z databáze potřebný záznam o konkrétním reportu a poté vrátí zpět objekt reportu. Webový klient si poté znovu načte seznam položek reportu, a přepíše jej seznamem položek reportu, který obsahuje načtený objekt reportu. Uživateli se poté překreslí seznam položek reportu.

Sekvence alt6 popisuje způsob uložení nově vytvořeného reportu. Webový klient si načte celý seznam položek reportu a název reportu, který buď zvolil systém nebo jej pomocí sekvence alt1 uživatel upravil. Poté takto načtený seznam položek odešle na server, který se postará o uložení záznamu do databáze. V případě úspěchu webový klient obeznámí uživatele o úspěšném uložení záznamu.

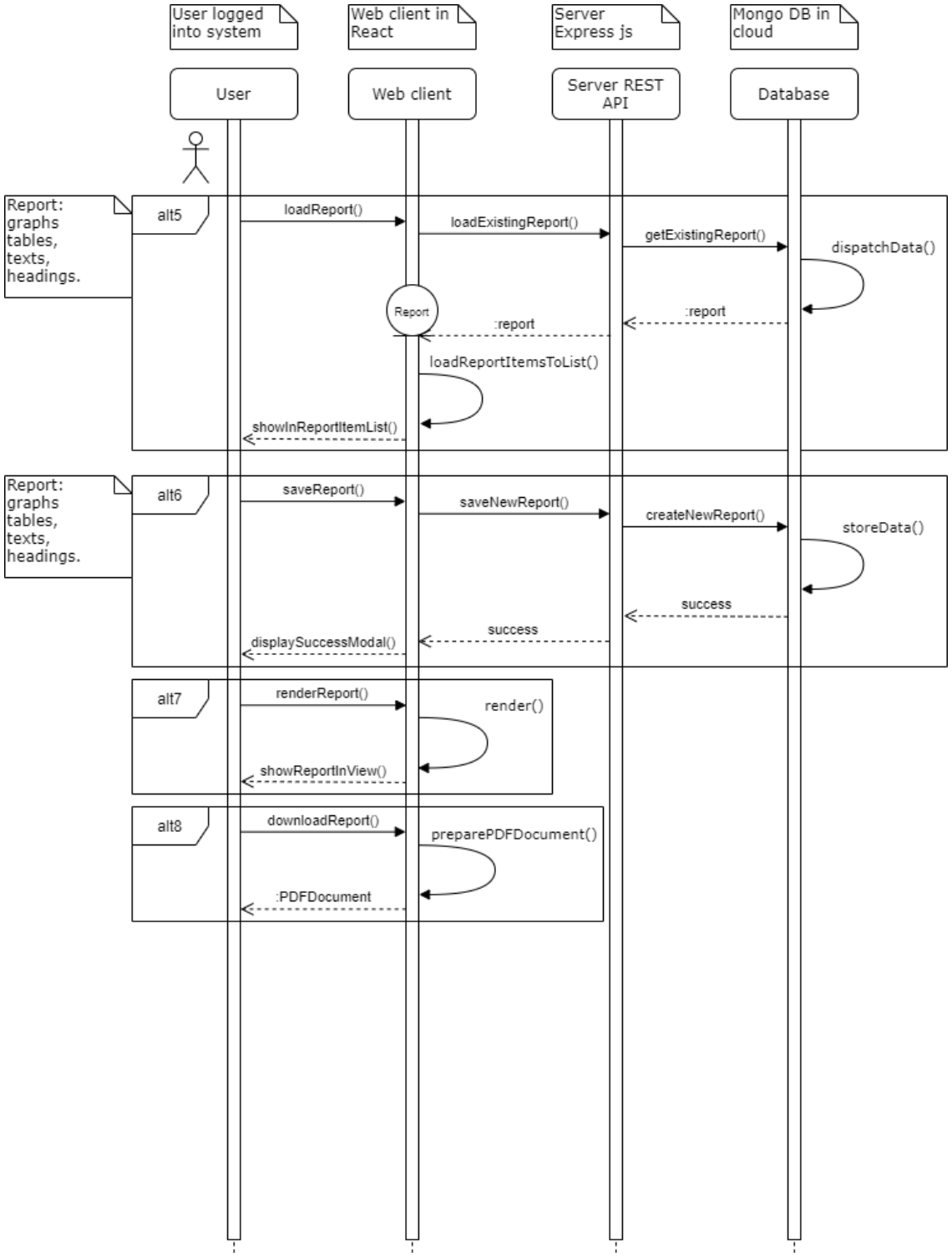
Sekvence alt7 znázorňuje funkci renderování reportu. Webový klient si načte všechny položky ze seznamu položek reportu a přemapuje je na dokument pdf, který poté vyrenderuje. Vyrenderovaný náhled pdf se zobrazí uživateli na stránce.

Alt8 je poslední sekvence druhého sekvenčního diagramu, která popisuje jakým způsobem uživatel stáhne požadovaný report ve formátu pdf. Webový klient připraví dokument a poté jej uživateli



Obrázek 4.5: Sekvenční diagram práce s reportovací částí systému

stáhne. Uživatel si může stažený dokument zobrazit nebo s ním dále pracovat.



Obrázek 4.6: Sekvenční diagram práce s reportovací částí systému

Kapitola 5

Implementace

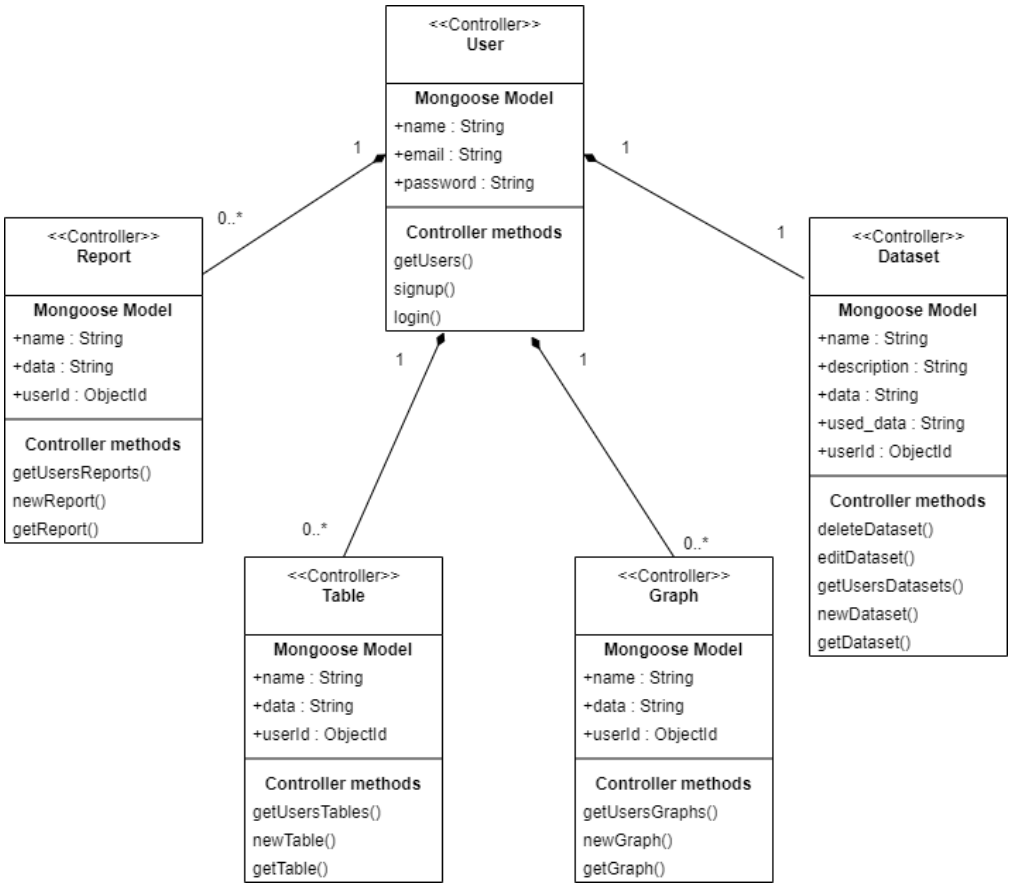
Kapitola implementace obsahuje popisy některých složitějších částí systému. Budou popsány složitější metody, diagramy jednotlivých tříd a implementační techniky, které stojí za zmínění. Nebudou detailně rozebírány použité technologie, ale spíše budou popsány v souvislosti s nějakým určitým naimplementovaným prvkem systému.

5.1 Server

V kapitole Architektura byly popsány použité technologie a obecné rozvržení do logických softwarových komponent. Následující obrázek 5.1 třídního diagramu zobrazuje detailnější pohled na tyto jednotlivé komponenty, které utváří middleware[31] server jako celek. Schéma se skládá z jednotlivých tříd, které jsou reprezentovány controllerem, jenž využívá modely knihovny mongoose[29].

Jak již bylo popsáno, serverová část je složena z modelů, controllerů a routerů. Modely jsou vytvořeny pomocí javascriptové knihovny mongoose, která slouží k přímému modelování dat aplikace. Mongoose vytváří jednotlivé databázové kolekce automaticky s prvním vloženým záznamem. Zahrnuje datové typy, validace dat a vytváření dotazů přímo do databáze. Knihovna mongoose zahrnuje metodu connect, do které se vloží řetězec pro připojení na databázi. Po zavolání této metody se naváže spojení s databází. S jednotlivými modely se v controllerech pracuje tak, že se naimportuje schéma, které bylo vytvořené v samostatném souboru. Toto schéma se poté chová jako objekt, který zpřístupňuje vybrané metody pro provádění operací nad daty. V systému existuje 5 modelů. Modely jsou popsány na obrázku 5.1.

Controllerů je stejný počet jako modelů. Každý model je pevně spjat s jedním konkrétním controllerem. Níže budou popsány implementace jednotlivých controllerů. Pro zjednodušení budou vynechány controllery pro tabulky a grafy, které jsou implementovány podobně jako například controller pro reporty. Seznam vybraných controllerů a popis jejich jednotlivých metod:



Obrázek 5.1: Třídní diagram serverové části

User Controller:

- metoda getUsers() - tato asynchronní metoda vrací seznam všech uživatelů v systému. Metoda využívá modelu User, na kterém volá metodu find(). Metoda find() bez vstupních parametrů vrátí všechny záznamy z databáze požadované kolekce. Záznamy jsou poté zaslány na výstup.

- metoda signup() - asynchronní metoda, která na vstupu přijímá name, email, password. Nejprve tyto vstupní parametry validuje. Pokud validace selže, vrátí se objekt se zprávou obsahující informace o špatně zadaných vstupech. Metoda dále z těchto vstupních parametrů sestaví objekt User. Objekt User je pak pomocí modelu User a voláním metody modelu save() uložen do databáze.

- metoda login() - asynchronní metoda zajišťující přihlášení uživatele. Na vstupu očekává email a password. Nejprve pomocí metody findOne(email: email) nalezne uživatele s daným emailem. Pokud uživatel existuje, metoda ověří, zda heslo na vstupu sedí s tím, které bylo uloženo v databázi. V případě úspěchu vrací objekt, který má atribut login s hodnotou true a atribut user obsahující objekt User.

Dataset Controller:

- metoda deleteDataset() - asynchronní metoda, přijímající na vstupu ID daného datasetu. Pomocí modelu Dataset a jeho metody findByIdAndDelete() nalezne v databázi daný dataset a trvale jej z databáze smaže. Pokud nenalezne žádný dataset, vrátí objekt s chybovou hláškou.
- metoda editDataset() - tato asynchronní metoda očekává na vstupu parametry name, description, ID datasetu a used data. Všechny tyto parametry musí být na vstupu zadány. Poté metoda pomocí modelu Dataset a jeho metody findByIdAndUpdate() edituje existující záznam datasetu. Pokud žádný dataset nenalezne, vrátí metoda objekt s chybovou hláškou.
- metoda getUsersDatasets() - asynchronní metoda očekává na vstupu ID uživatele. Pomocí modelu Dataset a jeho metody find() nalezne všechny záznamy datasetů, které patří uživateli s daným ID. Vrací objekt, který obsahuje seznam všech těchto objektů typu dataset.
- metoda getDataset() - metoda, která je asynchronní přijímá na vstupu ID datasetu, pro který se má vrátit na výstup jeho celý objekt. Model datasetu zajistí pomocí metody find() nalezení tohoto objektu s daným ID.
- metoda newDataSet() - asynchronní metoda, která přijímá na vstupu name, description, ID uživatele a soubor formátu xlsx. Metoda přesune získaný soubor do public adresáře. Poté načte data ze souboru a pomocí metody convertExcelToJson extrahuje tato data a serializuje je do datového formátu JSON. Z načtených vstupů a serializovaných dat do JSONu vytvoří objekt typu dataset. Poté se využije metoda modelu datasetu save() pro uložení záznamu do databáze. Pokud kterýkoliv z výše popsaných kroků selže, pak se na výstup vrátí objekt s chybovou hláškou. V úspěšném případě pak metoda vrací objekt typu dataset.

Report Controller:

- metoda getUsersReport() - asynchronní metoda očekává na vstupu ID uživatele. Pomocí modelu Report a jeho metody find() nalezne všechny záznamy reportů, které patří uživateli s daným ID. Vrací objekt, který obsahuje seznam všech těchto objektů typu Report.
- metoda newReport() - metoda, která je asynchronní. Přijímá na vstupu name, data a user ID. Data obsahují seznam veškerých objektů, které byly do reportu vloženy v rámci webového klienta (text, heading, table, graph). Metoda ze vstupů vytvoří objekt. Pomocí metody save() modelu report se uloží do databáze nově vytvořený záznam z tohoto objektu. Pokud se vytvoří nový záznam, pak metoda vrátí objekt nově vytvořeného záznamu.
- metoda getReport() - metoda, která je asynchronní. Přijímá na vstupu ID reportu, pro který se má vrátit na výstup jeho celý objekt. Model reportu zajistí pomocí metody find() nalezení tohoto objektu s daným ID.

Na výpisu ze zdrojového kódu 5.1 je vidět ukázka naimplementovaného modelu pomocí knihovny mongoose.

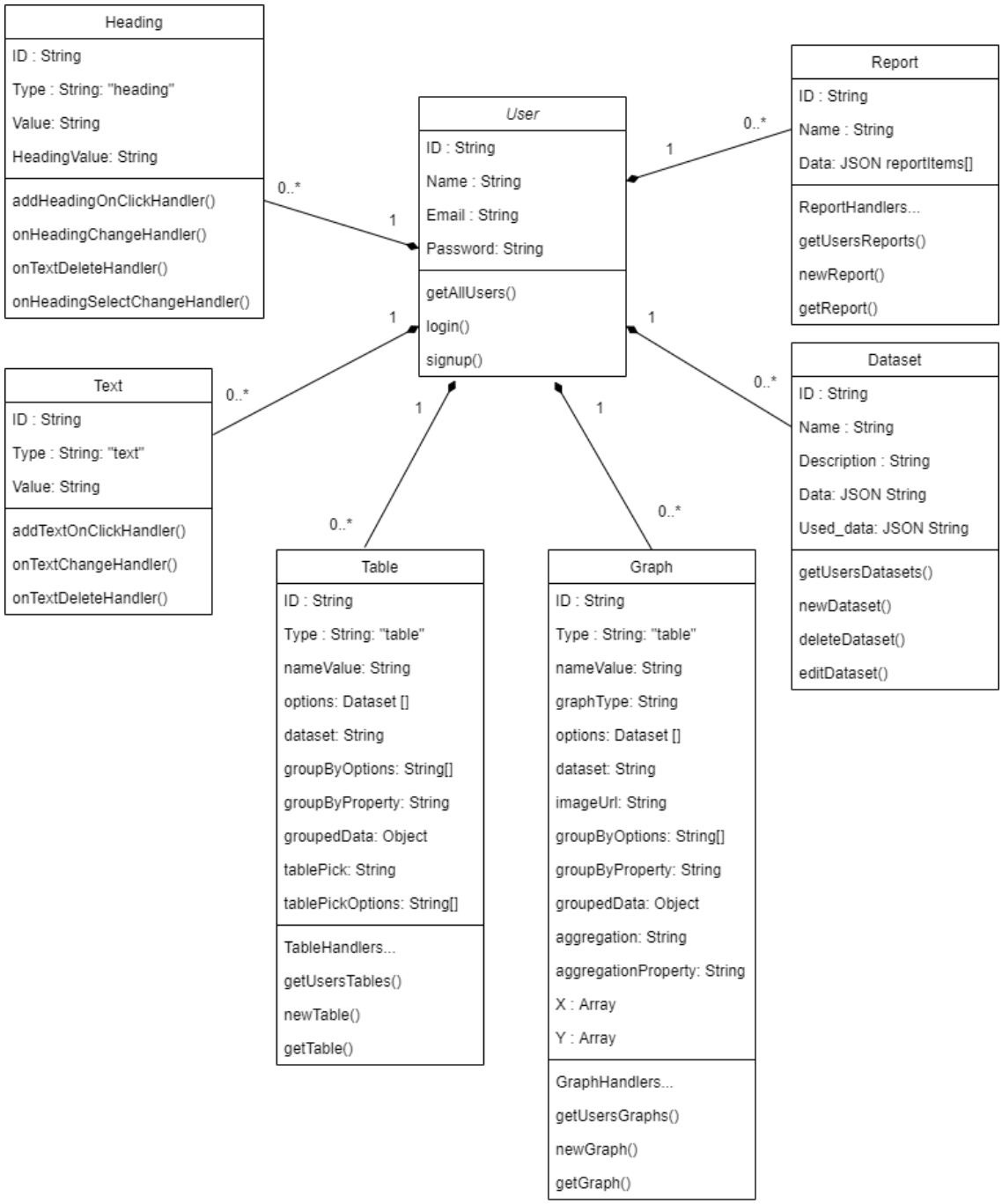
```
const mongoose = require("mongoose");

const dataSetSchema = new mongoose.Schema({
  name: {
    type: String,
  },
  description: {
    type: String,
  },
  data: {
    type: String,
  },
  used_data: {
    type: String,
  },
  userId: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'User'
  },
});

const DataSet = mongoose.model("DataSet", dataSetSchema);

module.exports = DataSet;
```

Listing 5.1: Ukázka implementace modelu mongoose.

5.2 Klient

5.2.1 Entity

Během implementace webového klienta byly dodrženy veškeré návrhy a postupy, které byly definovány v architektuře. Na obrázku 5.2 je zobrazen třídní diagram. Každá třída vyobrazena na diagramu se skládá z atributů dané entity a funkcí, které manipulují s těmito atributy a provádějí různé akce.

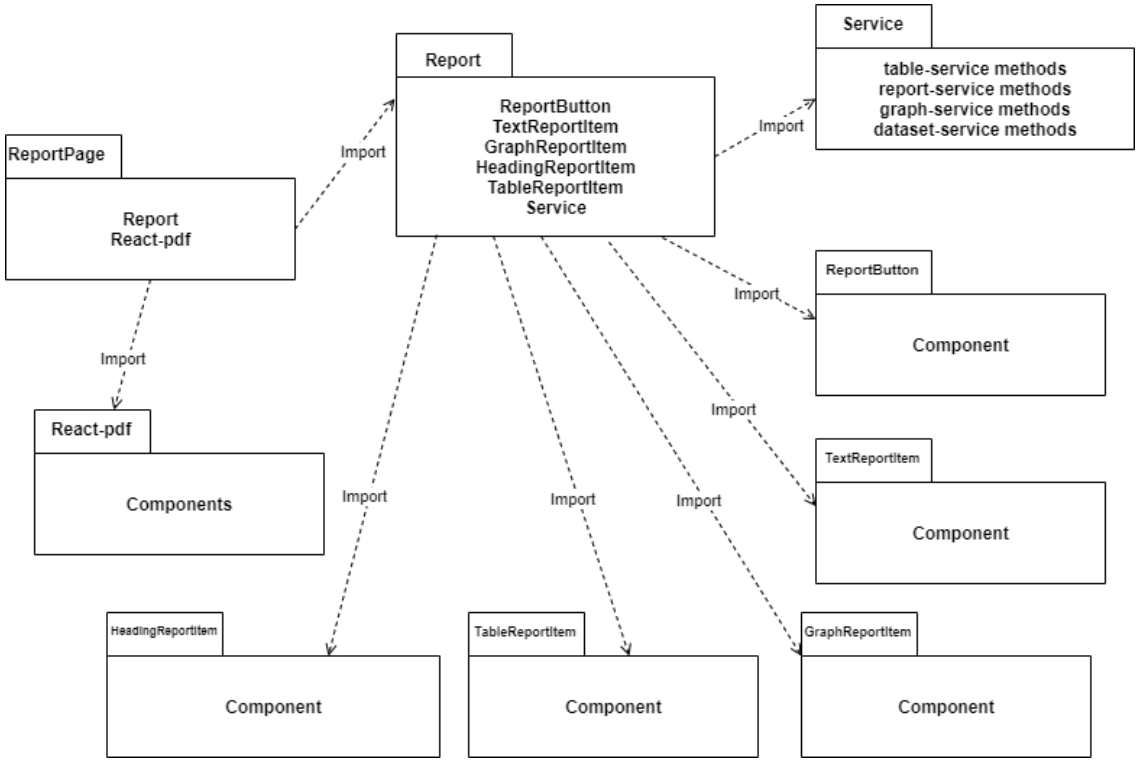


Obrázek 5.2: Třídní diagram webového klienta

5.2.2 Komponenty

Na obrázku 5.3 je zobrazen diagram komponent, který zobrazuje jednotlivé závislosti mezi React komponentami v aplikační části pro reportování. Vlastní definované React komponenty byly naimplementovány použitím funkčních komponent knihovny React. Tyto komponenty využívají sadu

takzvaných hooků, které zajišťují logiku překreslování jednotlivých komponent ve webové aplikaci. Nejvíce využitými hooky jsou useState() a useEffect(). Hook useState() uchovává stav objektu, případně pole objektů, nebo pouze nějakou hodnotu, kde se při její změně zajistí zavolání sekce komponenty return, která vrací obsah dané komponenty. Obsahem jsou HTML prvky nebo jiné komponenty, které se mají ve webovém klientovi zobrazit.



Obrázek 5.3: Diagram komponent reportovací části aplikace

Diagram na obrázku 5.3 obsahuje tyto komponenty:

- ReportPage - Komponenta, která importuje komponentu React-pdf[32] a komponentu Report.
- Report - Tato komponenta importuje ReportButton, TextReportItem, GraphReportItem, HeadingReportItem, TableReportItem a Service.
- Service - Komponenta obsahuje několik služeb. Každá služba definuje metody, které slouží pro získávání dat z backendu. Služby jsou následující: table-service, graph-service, dataset-service a report-service.
- ReportButton - Komponenta pro zobrazování tlačítek.
- TextReportItem - Komponenta pro práci s textovou položkou reportu.
- GraphReportItem - Komponenta pro práci s grafovou položkou reportu.

- TableReportItem - Komponenta pro práci s tabulkovou položkou reportu.
- HeadingReportItem - Tato komponenta slouží pro práci s nadpisy.
- React-pdf - Komponenta, která se stará o renderování položek do pdf formátu. Obsahuje také komponentu PDFViewer, která se stará o zobrazení náhledu pdf dokumentu ve webovém klientovi.

5.2.3 Implementace reportovací části webové aplikace

Reportovací část aplikace, jak již bylo řečeno, je sestavena pomocí stránky ReportPage. Stránka ReportPage obsahuje komponentu Report a komponentu PDFViewer. Komponenta stránky ReportPage definuje tyto metody:

- onRenderingHandler() - metoda, která je volána z komponenty Report. Slouží jako callback při stisknutí tlačítka Render v komponentě Report. Má za úkol vzít z komponenty Report seznam položek reportu (reportItems) a uložit je do svého stavu.
- documentMapper() - metoda, která mapuje objekty typu Text, Heading, Table a Graph do komponenty třídy document. Komponenta třídy dokument je vyrenderována pomocí komponenty PDFViewer a zobrazena v náhledu výsledného pdf reportu. Pro každý typ objektu je použit specifický způsob mapování. Text a nadpis využívají komponentu Text z knihovny react-pdf. Graf využívá komponentu Image také z knihovny react-pdf. Při mapování grafu se do komponenty Image vloží url adresa obrázku, která byla získána pomocí QuickChart. Objekt tabulky je mapován zvlášť pomocí komponent z knihovny react-pdf-table.

Komponenta Report zahrnuje téměř veškerou logiku tvorby objektů v reportu. Veškeré položky ze seznamu položek reportu jsou mapovány pomocí metody reportItemMapper() na vizuální prvky, které jsou viditelné pro uživatele ve webové aplikaci. Níže je uveden seznam všech typů metod, které jsou v komponentě Report naimplementovány. Metody, které implementují chování tabulek a grafů budou, vzhledem k jejich složitosti, popsány zvlášť.

REPORT

- onSaveReportClickHandler() - Definuje chování při kliknutí na tlačítko uložit report.
- onReportNameChangeHandler() - Definuje chování při změně názvu reportu.
- onReportLoadHandler() - Definuje chování při načtení konkrétního reportu z perzistentního úložiště.
- cancelReportModalHandler() - Definuje chování při zavření modálního okna pro načtení reportu.

- onLoadReportClickHandler() - Definuje chování po kliknutí na tlačítko načtení reportu.
- onReportSelectChangeHandler() - Definuje chování po výběru reportu, který se má načíst z perzistentního úložiště.
- cancelSaveModalhandler() - Definuje chování po zavření modálního okna zobrazující informativní zprávu o úspěšném uložení reportu.

TEXT

- addTextOnClickHandler() - Zajistí vložení nového objektu typu Text do seznamu položek reportu.
- onTextChangeHandler() - Definuje chování při změně textového pole v položce reportu typu Text.
- onTextDeleteHandler() - Definuje chování pro odebrání vybraného objektu typu Heading ze seznamu položek reportu.

HEADING

addHeadingOnClickHandler() - Zajistí vložení nového objektu typu Heading do seznamu položek reportu.

- onHeadingChangeHandler() - Definuje chování při změně textového pole v položce reportu typu Heading.
- onHeadingDeleteHandler() - Definuje chování pro odebrání vybraného objektu typu Heading ze seznamu položek reportu.
- onHeadingSelectChangeHandler() - Definuje chování při změně typu nadpisu v objektu typu Heading.

5.2.4 Popis metod pro práci s tabulkou

Metody, které definují chování objektu typu Table v uživatelském rozhraní webové aplikace.

Metoda addTableOnClickhandler()

Metoda definuje chování, které má za úkol přidat objekt typu Table do seznamu položek reportu. Přidává atributy: ID, type, nameValue, options, dataset, groupByOptions, groupByProperty, groupedData, tablePick a tablePickOptions.

Metoda onTableNameChangeHandler()

Metoda definuje chování při změně názvu objektu tabulky.

Metoda onTableDataSetChangeHandler()

Metoda definuje chování při změně datasetu, se kterým má objekt tabulky pracovat.

Metoda onTableDeleteHandler()

Metoda definuje chování při kliknutí na tlačítko odebrání konkrétní položky tabulky ze seznamu položek reportu.

Metoda onTableSaveHandler()

Metoda definuje chování při kliknutí na tlačítko uložení tabulky do databáze. Metoda načte data pro konkrétní tabulku ze seznamu položek reportu. Tato data uloží pomocí volání metody newTable() služby table-service.

Metoda cancelTableSaveModalHandler()

Metoda zavře modální okno, které zobrazovalo hlášku o úspěšném uložení záznamu tabulky do databáze.

Metoda addExistingTableOnClickHandler()

Metoda otevře modální okno, ve kterém zobrazí ve výběrovém seznamu všechny tabulky, které měl uživatel doposud uloženy.

Metoda cancelTableLoadModalHandler()

Metoda zavře modální okno, které bylo vyvoláno metodou addExistingTableOnClickHandler().

Metoda onTableLoadHandler()

Metoda získá objekt typu Table ze serveru podle ID tabulky, kterou si uživatel zvolil v seznamu svých tabulek během operace přidání existující tabulky.

Metoda onTableSelectChangeHandler()

Metoda, která se volá při výběru tabulky ze seznamu tabulek pro načtení z databáze. Uloží do stavu komponenty report hodnotu ID vybrané tabulky.

Metoda onTableGroupByChangeHandler

Metoda, která zajistí shlukování dat zvoleného datasetu dle zvoleného atributu. Shlukování dat probíhá pomocí metody groupBy() z knihovny Lodash.

Metoda onTableGroupByChangeHandler

Metoda starající se o to, která tabulka se po použití groupBy vykreslí. Možností je zvolit buď all, tedy všechny tabulky, nebo jednu z konkrétních tabulek, které jsou pojmenovány dle shlukovaných skupin.

5.2.5 Popis metod pro práci s grafem

Metody, které definují chování objektu typu Graph v uživatelském rozhraní webové aplikace.

Metoda addGraphOnClickHandler()

Metoda definuje chování, které má za úkol přidat objekt typu Graph do seznamu položek reportu. Přidává atributy: ID, type, nameValue, options, dataset, imageUrl, groupByOptions, groupByProperty, groupedData, aggregation, aggregationProperty, X a Y.

Metoda onGraphNameChangeHandler()

Metoda definuje chování při změně názvu objektu grafu.

Metoda onGraphDataSetChangeHandler()

Metoda definuje chování při změně datasetu, se kterým má objekt grafu pracovat.

Metoda onGraphDeleteHandler()

Metoda definuje chování při kliknutí na tlačítko odebrání konkrétní položky grafu ze seznamu položek reportu.

Metoda onGraphTypeChangeHandler()

Tato metoda zajišťuje změnu typu grafu, pokud uživatel zvolil mezi grafem sloupcovým a koláčovým.

Metoda onGraphUrlChangeHandler()

Metoda nastaví konkrétnímu objektu typu Graph url adresu obrázku, která obsahuje vykreslený graf.

Metoda onGraphGroupByChangeHandler()

Metoda, která zajistí shlukování dat zvoleného datasetu dle zvoleného atributu. Shlukování dat probíhá pomocí metody groupBy() z knihovny Lodash.

Metoda onGraphSaveHandler()

Metoda definuje chování při kliknutí na tlačítko uložení grafu do databáze. Metoda načte data pro konkrétní graf ze seznamu položek reportu. Tato data uloží pomocí volání metody newGraph() služby graph-service.

Metoda cancelGraphSaveModalHandler()

Metoda zavře modální okno, které zobrazovalo hlášku o úspěšném uložení záznamu grafu do databáze.

Metoda addExistingGraphOnClickHandler()

Metoda otevře modální okno, ve kterém zobrazí ve výběrovém seznamu všechny grafy, které měl uživatel doposud uloženy.

Metoda onGraphLoadHandler()

Metoda získá objekt typu Graph ze serveru podle ID grafu, kterou si uživatel zvolil v seznamu svých grafů během operace přidání existujícího grafu.

Metoda graphBarMapper()

Metoda zjistí, jestli byla zvolena možnost groupBy. Pokud nebyla zvolena možnost groupBy, pak nastaví konfiguraci grafu tak, že vezme hodnoty zvoleného X a hodnoty zvoleného Y. Poté odešle skrze REST API na službu QuickChart[27] požadavek na vykreslení grafu. QuickChart odpoví url adresou obrázku vyrenderovaného grafu. Nastaví url adresu danému objektu grafu pomocí metody onGraphUrlChangeHandler(). Pokud byla zvolena možnost groupBy, metoda shlukuje data do skupin dle uživatelem vybraného atributu. Toto je zajištěno pomocí metody groupBy z knihovny Lodash. Poté si načte uživatelem zvolenou agregační funkci (sum - sumace, avg - průměr). Provede agregaci nad hodnotami v dané skupině vybraného atributu datasetu. Poté nastaví jako popisky osy X názvy jednotlivých skupin, které vznikly při shlukování. Do popisků Y přidá agregovaná data pro jednotlivé skupiny. Poté odešle skrze REST API na službu QuickChart požadavek na vykreslení grafu. QuickChart odpoví url adresou obrázku vyrenderovaného grafu. Nastaví url adresu danému objektu grafu pomocí metody onGraphUrlChangeHandler().

Metoda onGraphBarXChangeHandler()

Metoda, která se volá při výběru atributu datasetu, který bude zvolen jako popisek osy X grafu. Tato možnost je dostupná pouze pokud uživatel nevybral možnost groupBy. Metoda si načte dataset a poté zavolá funkci graphBarMapper().

Metoda onGraphBarYChangeHandler()

Metoda, která se volá při výběru atributu datasetu, který bude zvolen jako popisek osy Y grafu. Tato možnost je dostupná pouze pokud uživatel nevybral možnost groupBy. Metoda si načte dataset a poté zavolá funkci graphBarMapper().

Metoda onGraphBarAggregationChangeHandler()

Metoda načte uživatelem zvolenou agregační funkci a poté zavolá metodu graphMapper().

Metoda onGraphBarAgrPropertyChangeHandler()

Metoda načte uživatelem zvolený atribut, který bude agregován a poté zavolá metodu graphMapper().

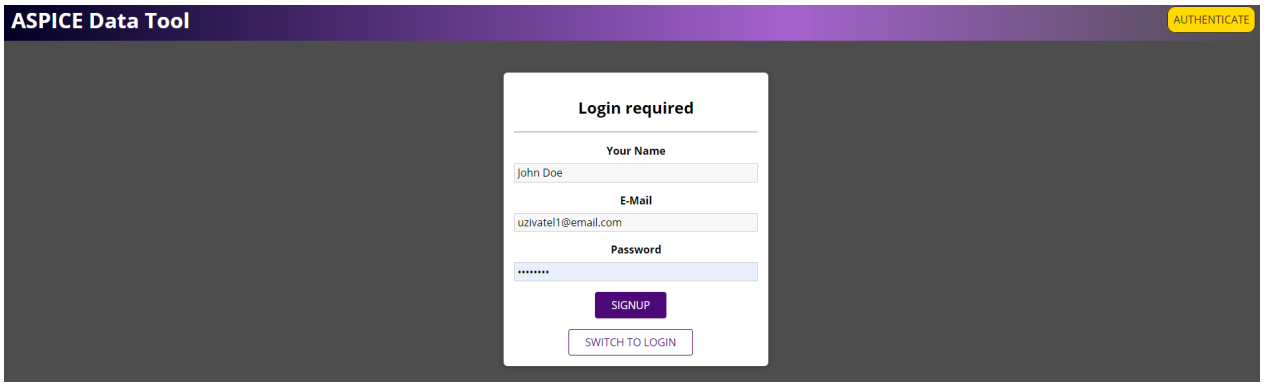
Kapitola 6

Výsledná webová aplikace

Výsledná webová aplikace včetně naimplementované serverové strany slouží uživatelům k nahrání svých dat ve formátu xlsx do formy datasetu, který potom mohou upravovat pomocí funkce editace datasetu. Uživatel vybírá sloupce (atributy), které chce využívat při tvorbě reportu. Reportovací část webové aplikace slouží k vytváření celkových reportů pomocí prvků textu, grafu, tabulky a nadpisu. Systém byl navržen a naimplementován tak, aby mohl být využíván nejen hodnotiteli ASPICE procesů, ale kýmkoli, kdo si nahraje do systému zdrojová data ve formátu xlsx.

Přihlášení a registrace

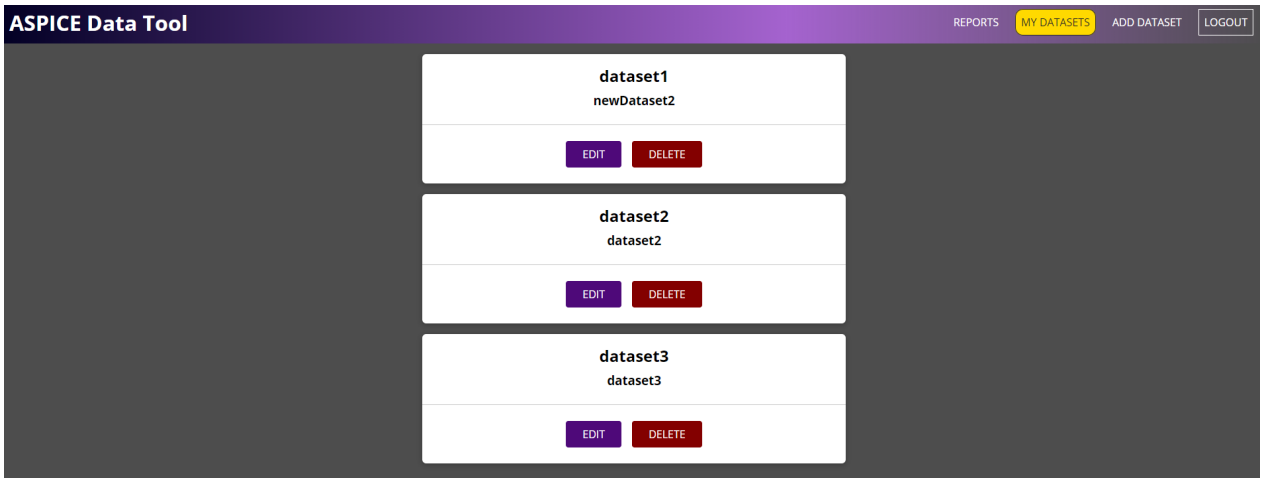
Obrázek 6.1 představuje ukázku registrace nového uživatele a přihlášení.



Obrázek 6.1: Přihlášení a registrace do systému.

Seznam datasetů uživatele

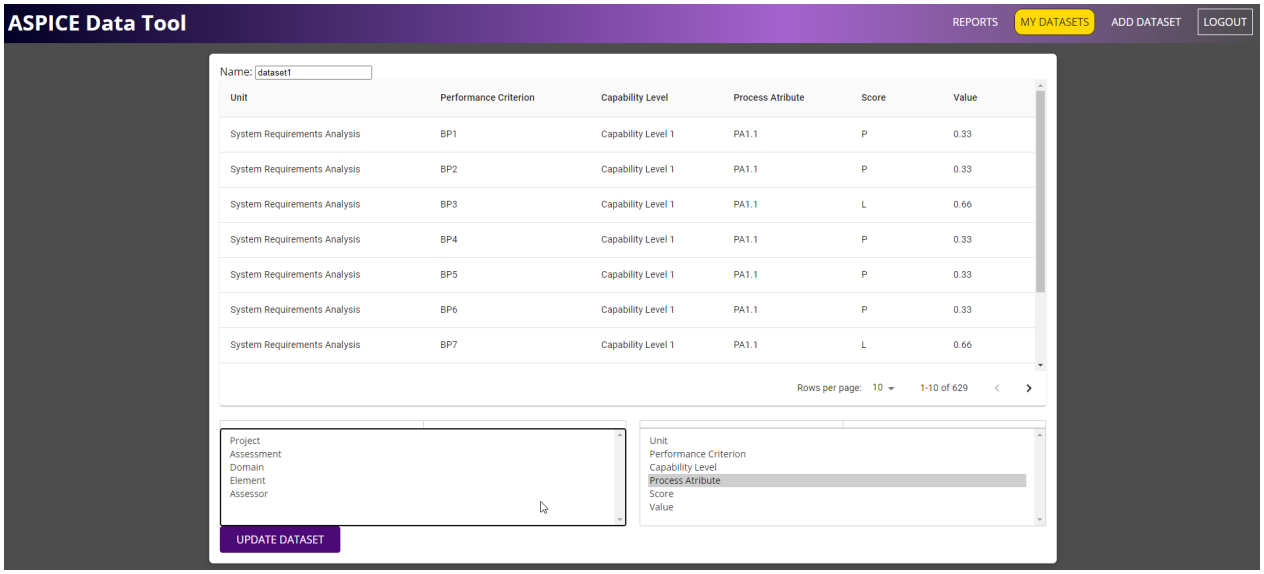
Obrázek 6.2 představuje ukázku, kde si uživatel může zobrazit seznam svých datasetů. V této části může zvolit editaci nebo smazání konkrétního datasetu.



Obrázek 6.2: Seznam datasetů uživatele.

Editace datasetu

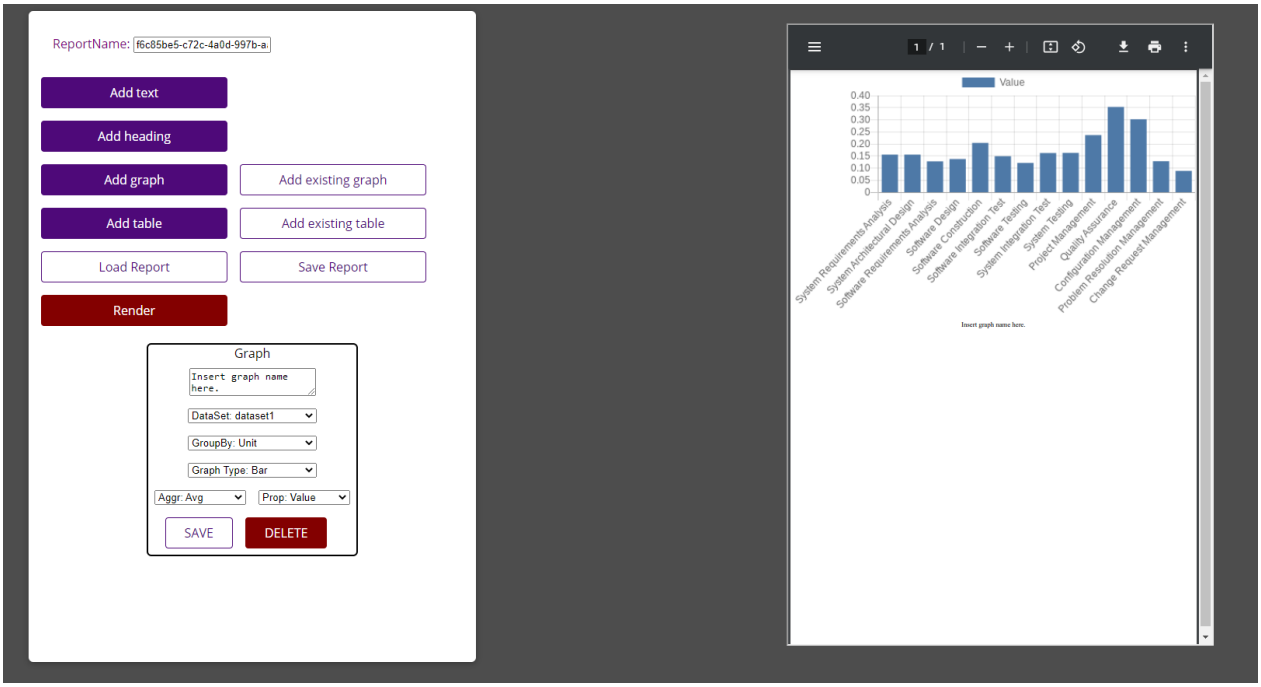
Následující obrázek 6.3 zobrazuje část webové aplikace, kde si uživatel upravuje svůj dataset. Pomocí dvou seznamů přetahuje uživatel na pravou stranu ty atributy, které chce aby byly použity.



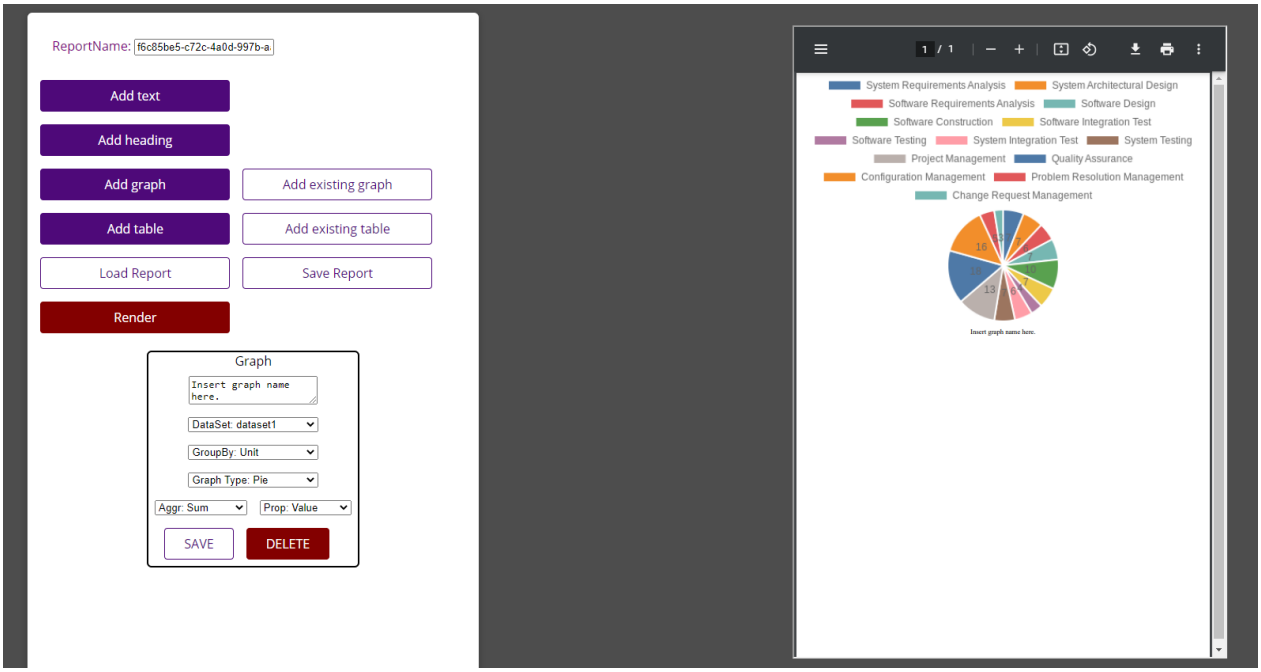
Obrázek 6.3: Editace datasetu.

Grafy

Na obrázku 6.4 lze vidět vykreslený sloupcový graf, zatímco koláčový graf je zobrazen na obrázku 6.5.



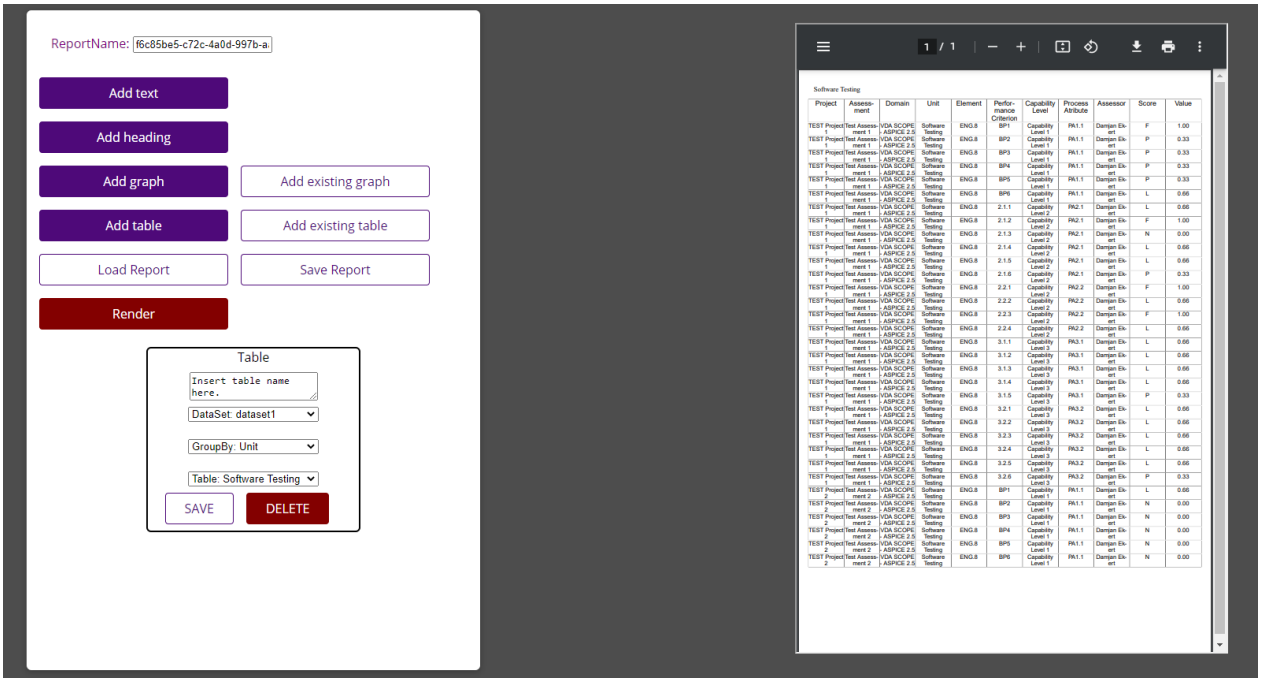
Obrázek 6.4: Sloupcový graf.



Obrázek 6.5: Koláčový graf.

Tabulky

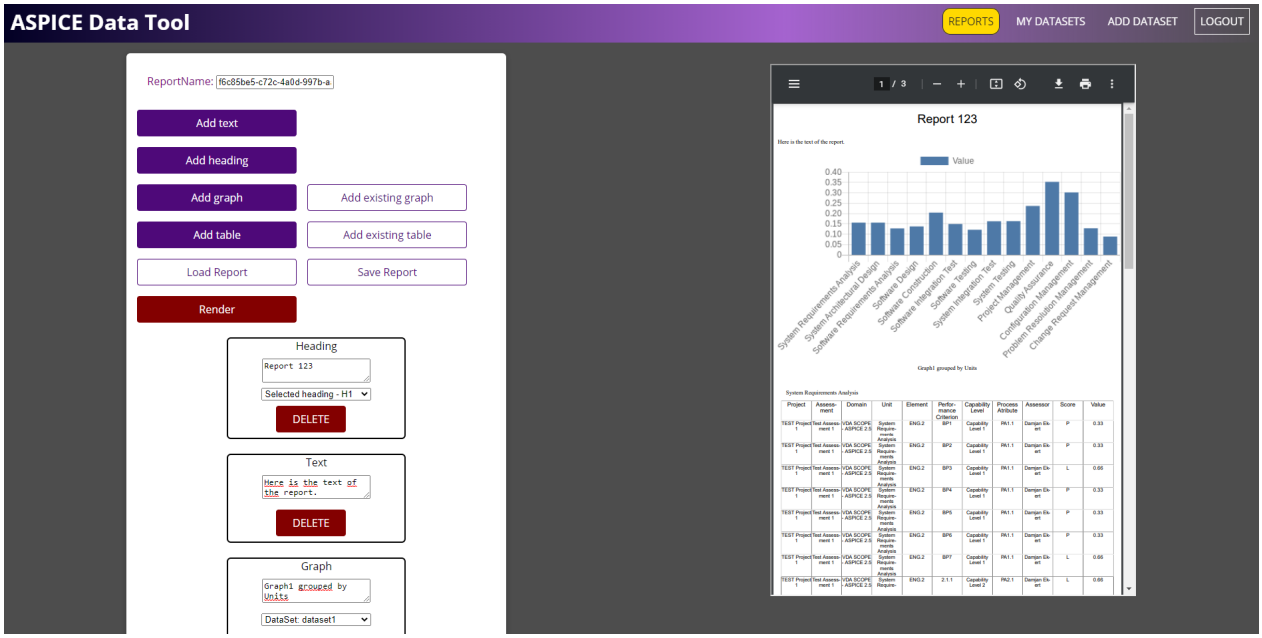
Obrázek 6.6 ukazuje vykreslenou tabulku, která byla vytvořena na základě volby parametrů objektu tabulky.



Obrázek 6.6: Tabulka.

Vytvoření reportu

Obrázek 6.7 znázorňuje práci s reportovací částí webové aplikace. V levé části jsou tlačítka, pomocí kterých uživatel pracuje s reportem a seznamem položek reportu. Tlačítko pro rednerování umožní vykreslit výsledný report, který se zobrazí jako náhled v pravé části. Poté jej uživatel může stáhnout ve formátu pdf pomocí tlačítka stáhnout. Na zmíněném obrázku 6.7 je tedy viditelný report, který se skládá z nadpisu, textu, grafu a tabulky. Nadpis byl nastavený s velikostí H1, grafu byl tvořen shlukováním pomocí groupBy a atributu Unit z testovacího datasetu. Poté byla zvolena agregační funkce pro průměr, která se provedla nad atributem Value z testovacího datasetu. Do výsledného grafu pak byly naneseny na popisky osy X hodnoty názvů jednotlivých shluků. Na osu Y byly naneseny hodnoty atributu Value, které byly zprůměrovány v rámci své skupiny. Pro zobrazenou tabulku byla zvolena, stejně jako u grafu, také možnost groupBy dle atributu Unit z testovacího datasetu. Poté bylo zvoleno vykreslení tabulky pro skupinu System Requirements Analysis, která vznikla jako jedna ze skupin z atributu Unit.



Obrázek 6.7: Tvorba reportu.

Kapitola 7

Závěr

Výsledkem této práce bylo vytvoření webová aplikace, která slouží pro generování uživatelem definovaných reportů. Původním záměrem aplikace byl poskytnout hodnotitelům procesů v oblasti Automotive nástroj, který by jim umožňoval naimportovat data a poté z těchto dat vytvářet vlastní reporty. Během implementace však došlo ke generalizaci některých prvků a komponent, díky čemuž může vytvořená aplikace sloužit k práci s daty různých odvětví. Během fáze návrhu aplikace a architektury byla kladena otázka jaké technologie zvolit.

Autor práce měl v úmyslu nastudovat modernější technologie, které by bylo možné využít právě při implementaci reportovacího nástroje. Tyto technologie byly nastudovány, použity a popsány v kapitolách architektury a implementace.

Na samotném začátku práce byla popsána norma Automotive SPICE, jež nastiňuje problematiku vývoje reportovacího nástroje pro hodnotitele procesů. Součástí tohoto popisu bylo představení tří komerčních nástrojů, které bohužel nebyly volně dostupné, avšak byly nastudovány z oficiálních stránek výrobců těchto nástrojů.

Vedoucím práce byl poskytnut soubor ve formátu xlsx, jenž obsahuje výstupní data hodnocení ze zmíněného nástroje Capability Adviser. Tato data byla použita jako testovací data během vývoje aplikace. Ukázky výsledného systému je možno vidět v kapitole 6.

Během implementace webové aplikace vznikl problém, kdy bylo nutno propojit několik technologií. Zvolená knihovna react-pdf, která umožňuje tvorbu vlastního PDF dokumentu, neobsahovala nativně žádnou možnost, jak zajistit vykreslování tabulek. Nakonec však byla nalezena dodatečná knihovna[33], která tuto možnost zpřístupňuje. Dále vznikla otázka, jakým způsobem se do reportu budou generovat grafy. Řešení přinesla veřejně dostupná služba QuickChart, která je schopna na základě definovaných vstupů pomocí REST API vrátit url adresu s vykresleným grafem.

Reportovací aplikace může být v budoucnu využita a dále rozšiřována. Dodatečnou funkcí by mohlo být zpřístupnění uživateli rozsáhlejších funkcí pro výběr dat při vytváření grafů a tabulek.

Literatura

1. LOON, Han van. *Process Assessment and ISO/IEC 15504. A reference book. (2nd edition in Englisch)*. 1. vyd. Springer, 2007.
2. HÖHN, Holger; SECHSER, Bernhard; DUSSA-ZIEGER, Klaudia; MESSNARZ, Richard; HINDEL, Bernd. *Software Engineering nach Automotive SPICE. Entwicklungsprozess in der Praxis - Ein Continental-Projekt auf dem Weg zu Level 3. dpunkt Verlag*. 1. vyd. Heidelberg, 2009.
3. HÖRMANN, Klaus; DITTMANN, Lars; HINDEL, Bernd; MÜLLER, Markus. *SPICE in der Praxis. Interpretationshilfe für Anwender und Assessoren. dpunkt Verlag*. 1. vyd. Heidelberg, 2006.
4. *ISO* [online] [cit. 2021-04-10]. Dostupné z: `https://www.iso.org/committee/45086/x/catalogue/`.
5. MÜLLER, Markus; HÖRMANN, Klaus; DITTMANN, Lars; ZIMMER, Jörg. *Automotive SPICE in der Praxis. Interpretationshilfe für Anwender und Assessoren. dpunkt Verlag*. 1. vyd. Heidelberg, 2007.
6. LOON, Han van. *Process Assessment and Improvement. A practical guide. (2nd edition in Englisch)*. 1. vyd. Springer, 2007.
7. *ASPICE* [online] [cit. 2021-04-10]. Dostupné z: `https://beza1e1.tuxen.de/aspice.html`.
8. *Clarus: Concept of Operations* [online] [cit. 2021-04-10]. Dostupné z: `https://web.archive.org/web/20090705102900/http://www.itsdocs.fhwa.dot.gov/jpodocs/repts_te/14158.htm`.
9. *ASPICE* [online] [cit. 2021-04-10]. Dostupné z: `http://www.automotivespice.com/fileadmin/software-download/AutomotiveSPICE_PAM_31.pdf`.
10. *CapabilityAdviser* [online] [cit. 2021-04-25]. Dostupné z: `http://www.iscn.com/capadv/#products`.
11. *Capability Adviser ISCN* [online] [cit. 2021-04-20]. Dostupné z: `http://www.iscn.com/capadv/images/Capability_Adviser_An_Introduction.pdf`.
12. *SPiCE 1-2-1* [online] [cit. 2021-04-25]. Dostupné z: `http://www.spice12drive.com/cms/en/about-spice-1-2-1.html`.

13. *Callis Trace* [online] [cit. 2021-04-25]. Dostupné z: http://www.callis.dk/callis-trace/.

14. *MOCKFLOW* [online] [cit. 2021-04-15]. Dostupné z: https://mockflow.com/.

15. *Client, Server Architecture* [online] [cit. 2021-04-20]. Dostupné z: https://cio-wiki.org/wiki/Client_Server_Architecture.

16. *Peer-to-Peer* [online] [cit. 2021-04-20]. Dostupné z: https://www.computerworld.com/article/2588287/networking-peer-to-peer-network.html.

17. *MVC IT Network* [online] [cit. 2021-04-25]. Dostupné z: https://www.itnetwork.cz/navrh/mvc-architektura-navrhovy-vzor.

18. *What-is-a-fat-client* [online] [cit. 2021-04-20]. Dostupné z: https://www.netinbag.com/cs/internet/what-is-a-fat-client.html.

19. *SOAP* [online] [cit. 2021-04-20]. Dostupné z: https://www.w3schools.com/xml/xml_soap.asp.

20. *Xml vs JSON* [online] [cit. 2021-04-21]. Dostupné z: https://www.guru99.com/json-vs-xml-difference.html.

21. *React* [online] [cit. 2021-04-21]. Dostupné z: https://reactjs.org/.

22. *Express js* [online] [cit. 2021-04-21]. Dostupné z: https://expressjs.com/.

23. *Javascript* [online] [cit. 2021-04-21]. Dostupné z: https://www.w3schools.com/js/DEFAULT.asp.

24. *Node js* [online] [cit. 2021-04-21]. Dostupné z: https://nodejs.org/en/about/.

25. *Mongo vs SQL* [online] [cit. 2021-04-21]. Dostupné z: https://www.knowi.com/blog/mongodb-vs-sql/.

26. *Mongo DB* [online] [cit. 2021-04-21]. Dostupné z: https://www.mongodb.com/2.

27. *quickchart* [online] [cit. 2021-04-21]. Dostupné z: https://quickchart.io/.

28. *REST API* [online] [cit. 2021-04-21]. Dostupné z: https://www.itnetwork.cz/programovani/nezarazene/stoparuv-pruvodce-rest-api/.

29. *Mongoose* [online] [cit. 2021-04-22]. Dostupné z: https://mongoosejs.com/docs/guide.html.

30. *ACID* [online] [cit. 2021-04-22]. Dostupné z: https://database.guide/what-is-acid-in-databases/.

31. *Middleware* [online] [cit. 2021-04-22]. Dostupné z: https://www.redhat.com/en/topics/middleware/what-is-middleware.

32. *React-pdf* [online] [cit. 2021-04-22]. Dostupné z: https://react-pdf.org/.

33. *React-pdf-table* [online] [cit. 2021-04-22]. Dostupné z: https://www.npmjs.com/package/@david.kucsai/react-pdf-table.

Příloha A

Nasazení aplikace

Aplikace využívá několik technologií, které musí být nainstalovány na server, kde aplikace poběží. Na serveru musí být nainstalováno běhové prostředí Node.js a spolu s ním správce balíčků npm. Server musí být připojen k internetu, aby mohl využít službu QuickChart.

Webový klient

V adresáři "/code/frontend"se nachází zdrojové kódy webové aplikace. Před jakoukoli následující akcí, musí být proveden příkaz "npm install"v adresáři "/code/frontend". Tento příkaz zajistí stažení a nainstalování všech potřebných balíčků. Jakmile jsou staženy veškeré potřebné balíčky, pak je nutno aplikaci zkompilovat pomocí příkazu "npm run build". Po úspěšné kompilaci se vytvoří adresář build, který obsahuje veškeré potřebné soubory pro spuštění webové aplikace. Spuštění aplikace zajistí balík serve, který je nutno nainstalovat pomocí příkazu "npm install -g serve". Aplikaci je možno spustit pomocí příkazu "serve -s build -l 3000". Po úspěšném spuštění bude webová aplikace dostupná na adrese «Adresa serveru>:3000". Adresu je možno zadat do webového prohlížeče, který webovou aplikaci načte.

Serverová aplikace

V adresáři "/code/backend"se nachází zdrojové kódy serverové aplikace. Před jakoukoliv následující akcí, musí být proveden příkaz "npm install"v adresáři "/code/backend". Jakmile jsou všechny potřebné balíčky staženy a nainstalovány, poté je možné serverovou aplikaci spustit pomocí příkazu "npm start". Ve výchozím nastavení serverová aplikace využívá cloudovou databází MongoDB, která byla vytvořena autorem této práce. Adresu databáze je možné změnit na čtvrtém řádku v souboru "/code/backend/database/database.js".